StarBat

for StarWarp®

Version 6.1.0

Order Number: STBUW610

---

serena

Contents

PREFACE ..... **I**

SOFTWARE ENVIRONMENT I
WHAT IS STARBAT? II
WHAT TYPES OF DATA SETS CAN BE PROCESSED? II
STARBAT INITIALIZATION II

INTRODUCTION **1**

GENERAL OVERVIEW 1
MAJOR FUNCTIONAL FEATURES 1

STARBAT CONCEPTS **3**

FUNCTIONS AND MODIFIERS 3
PARAMETER DESCRIPTIONS 4

STARBAT DATA SETS **5**

SEQUENTIAL DATA SETS 5
DIRECT DATA SETS 5
VSAM DATA SETS 5
PARTITIONED DATA SETS 5
STARBAT JCL AND DDNAMES 6

STARBAT CONTROL STATEMENTS **7**

DATA SET IDENTIFIER 7
FUNCTION IDENTIFIER 7
PARAMETERS 7

STARBAT FUNCTIONS **11**

COPYREC OR CR, COPYALL OR CA, COPYMBR OR CM, COPYSOME OR CS FUNCTION 12
COPYREV OR CPR FUNCTION 12
EXCLUDEREC OR XR FUNCTION 13
MULTICOPY OR MC FUNCTIONS 13
PRINT OR P, PRINTALL OR PA, PRINTMBR OR PM, PRINTREV OR PRR FUNCTIONS 14
PRINTCHR OR PC, PRINTCHRALL OR PCA, PRINTCHRMBR OR PCM, PRINTCHRREV OR PCR FUNCTIONS 14
PRINTHEX OR PH, PRINTHEXALL OR PHA, PRINTHEXMBR OR PHM, PRINTHEXREV OR PHR FUNCTIONS 15
SKIP OR S, SKIPREV OR SKR FUNCTION 15
TOTAL OR T FUNCTIONS 16
UPDATEREC OR UR, UPDATEMBR OR UM, UPDATEALL OR UA FUNCTIONS 16

STARBAT PARAMETERS **17**

ABEND OR AB PARAMETER 18
AND OR IF PARAMETER 18
CHANGE OR C PARAMETER 19
CHANGEALL OR CA PARAMETER 19
COPYOVER OR CO PARAMETER 20
EXCLUDEREC OR XR PARAMETER 20
EXPAND OR EX PARAMETER 20
IF OR AND PARAMETER 21
MAXDATERR PARAMETER 22

MAXRECIN OR MRI PARAMETER....22
MAXRECOUT OR MRO PARAMETER....22
MEMBER OR M PARAMETER....22
MEMBERS OR MS PARAMETER....23
MOVE OR MV PARAMETER....23
NEWMBR OR NM PARAMETER....24
NEWMBRS OR NMS PARAMETER....24
OPTIONS OR OP PARAMETER....24
OR PARAMETER....25
OVERLAY OR OL PARAMETER....26
OVERALL OR OA PARAMETER....27
PADCHAR OR PAD PARAMETER....27
PRINT OR P PARAMETER....27
PRINTCHR OR PC PARAMETER....28
PRINTHEX OR PH PARAMETER....28
PRINTLPI OR PL PARAMETER....28
RBA PARAMETER....28
RDW PARAMETER....29
SELECT OR S PARAMETER....29
STARTKEY OR SK PARAMETER....29
STOPIF OR ST PARAMETER....30
SUM PARAMETER....30
WARP PARAMETER....31
Warping Numeric Data....*31*
Warping Dates....*33*
WARPDEF OR WD PARAMETER....38
WRITE OR W PARAMETER....41

STARBAT LOGIC....43

LOGICAL AND CONDITIONS....43
LOGICAL OR CONDITIONS....43
PROCESSING MULTIPLE SELECTION PARAMETERS....43
SELECTING MEMBERS BY CONTENT....44
SELECTING MEMBERS BY NAME....44
OPTIONS=JCL PROCESSING....45

STARBAT EXAMPLES....47

STARBAT SAMPLE EXECUTION....51

STARBAT MESSAGES AND CODES....53

UNNUMBERED MESSAGES....53

INDEX....55

Preface

STARBAT is a part of the STARTOOL and STARWARP® products; it is a batch program that is normally used for repetitive, bulk data processing tasks. This manual contains detailed information on the use of STARBAT.

Some of STARBAT's features are:

- Records can be copied in whole or in part from any data set of any type to any other data set of the same or a different type
- Records can be printed in text or vertical hexadecimal
- Data items within records can be replaced with larger or smaller data items
- An entire data set can be processed at one time, or records can be processed selectively
- Totals can be created for particular data within the data set
- Data or date fields can be warped

Note: you will be able to use STARTOOL and STARWARP features only if you are licensed to use both products. Please contact SERENA Software sales if you want to arrange a trial of the other product.

Software Environment

STARBAT runs under two major IBM MVS operating systems:

- MVS/ESA *(any release)*
- OS/390 *(any release)*

In addition, the following environments should be available:

- ISPF and ISPF/PDF *(Version 4.0 or above)*
- TSO/E *(any release or any version)*

STARWARP

```
Version ..............  6
Release ..............  1
Modification .........  0
Julian release date ... 2000.001
```

What is STARBAT?

STARBAT is an MVS batch program using standard MVS JCL to manipulate data files. STARBAT supports data in character, hexadecimal, packed, and binary formats on disk or on tape.

Using STARBAT, you can copy, update, or print an entire file or specific records within the file that satisfy certain conditions. You can also use STARBAT to process numeric data, modify dates, and convert date formats.

What Types of Data Sets Can Be Processed?

STARBAT can process disk files with partitioned (PDS), partitioned extended (PDSE), sequential, VSAM, or direct organization; it can process tape files with sequential organization. Any record format is acceptable; STARBAT assumes that any partitioned data set with record format U is a load library and that other libraries are usually called "source" libraries.

STARBAT Initialization

When STARBAT is invoked, license information is normally displayed as shown below to identify the program, release number and release date.

```
PDS100I STARBAT/Both -- Version 6.1.0  2000.001

Proprietary software product of SERENA Software
Phone (650)696-1800 OR FAX (650)696-1776
LICENSED TO: your corporate name/trial offer expires ...
             your city, state, zip/agent to contact for license ...
All other rights reserved - use of this software
product by unauthorized persons is prohibited.
```

Figure 1. Sample License Information

Introduction

General Overview

In this introduction, we briefly describe the main functions that STARBAT provides. We introduce the major functions and explain how to combine the functions and parameters into a job to modify a data file.

Major functional features

There are two main uses for STARBAT.

- **Manipulate members of a data set**
  Different STARBAT functions can copy, print and update members of a data set. See COPYMBR, PRINTMBR, PRINTCHRMBR, PRINTHEXMBR, and UPDATEMBR for more information.

- **Manipulate records in a member or data set**
  This is by far the most common use of STARBAT. This allows for the copying, deleting, and inserting of records. Additionally, records can be printed as text, hexadecimal, or both. The data in the records can be totaled and reformatted in a variety of ways. The functions used here are COPYALL, COPYREC, COPYREV, COPYSOME, EXCLUDEREC, MULTICOPY, PRINT, PRINTALL, PRINTCHR, PRINTCHRALL, PRINTCHRREV, PRINTHEX, PRINHEXALL, PRINTHEXREV, PRINTREV, SKIP, SKIPREV, TOTAL, UPDATEALL, and UPDATEREC.

If we look at the functions themselves, they fall into five categories.

- **Copying data**
  There are many occasions when it is required to create a subset of a data file and, by combining the various copy functions, the exclude function and the conditional parameters, it is possible to develop sophisticated jobs to manipulate records to create new data files.

- **Deleting data**
  Whether it's deleting bad records or dropping records because they are no longer required, STARBAT has the ability to do this effortlessly. Again, in combination with conditional parameters, it is possible to edit a file in batch mode without having to create a special purpose program to do it.

- **Inserting data**
  In addition to copying and deleting records, STARBAT can insert new records if required. This allows new record types to be added, for example, to a test data file so that new functionality in a program can be tested.

- **Printing data**
  STARBAT allows for data sets and members to be printed in text mode, which is suitable for source files for example, and in hexadecimal mode, which is more suited to data files. This is invaluable in testing a program because it clearly shows the exact contents of the files.

- **Changing data**
  One of the most powerful features of STARBAT is its ability to modify the content of a record. Using commands similar to ISPF edit commands, it is possible to change whole files or individual records according to some defined pattern. These changes include the ability to increase the size of a field, as might be needed with converting a date format from MMDDYY to YYYYMMDD for Year 2000.

STARBAT Concepts

STARBAT is a program designed to perform a variety of data manipulation tasks in a background environment.

You can use STARBAT to copy selected records or portions of records from one data set type to another, print data in different formats, add or enlarge data fields in records, warp date fields, and process data set members selectively.

STARBAT is a MVS batch program that is controlled with standard MVS JCL. You provide control statements to specify the functions you want performed in your JCL or in a data set pointed to by the SYSIN DD statement.

For STARBAT, functions are defined to process data sets to your specifications. Parameters specified on these function statements limit the parts of the data set to be processed and control secondary processes.

Functions and Modifiers

Following is a brief description of each supported function.

Function	Short	Description
COPYALL	CA	Copies all records of a data set
COPYMBR	CM	Copies members conditionally based on contents
COPYREC	CR	Copies data and reports record totals
COPYREV	CPR	Copies data in reverse order
COPYSOME	CS	Copies selected records but applies all changes like COPYALL
EXCLUDEREC	XR	Eliminates unwanted records in a copy
MULTICOPY	MC	Copies data to one or more output data sets
PRINT	P	Prints records in alphanumeric format with record statistics
PRINTALL	PA	Prints all records of a data set
PRINTCHR	PC	Prints records in alphanumeric format
PRINTCHRALL	PCA	Prints all records of a data set
PRINTCHRMBR	PCM	Prints members conditionally based on contents
PRINTCHRREV	PCR	Prints records in alphanumeric format and reverse order
PRINTHEX	PH	Prints records in vertical hexadecimal format
PRINTHEXALL	PHA	Prints all records of a data set
PRINTHEXMBR	PHM	Prints members conditionally based on contents
PRINTHEXREV	PHR	Prints records in vertical hexadecimal format and reverse order
PRINTMBR	PM	Prints members conditionally based on contents
PRINTREV	PRR	Prints records in alphanumeric format and reverse order
SKIP	S	Moves the current record pointer forward
SKIPREV	SKR	Moves the current record pointer backward
TOTAL	T	Reads input records processing all parameter groups for SUM
UPDATEALL	UA	Updates all records of a data set
UPDATEMBR	UM	Updates members conditionally based on contents
UPDATEREC	UR	Updates records in place

Parameter Descriptions

Parameters are code words or keywords that control processing actions.

Following is a brief description of each supported parameter.

Parameter	Short	Description
ABEND	AB	Controls EOJ processing when abnormal condition occurs
AND	IF	Creates a logical AND condition check (used with IF)
CHANGE	C	Changes only the first instance of data in a record
CHANGEALL	CA	Changes all occurrences of data in a record
COPYOVER	CO	Controls the replacement of identically named output members
EXCLUDEREC	XR	Controls the # of records to bypass in an EXCLUDEREC function
EXPAND	EX	Expands records at a specified location
IF	AND	Selects records to process based on data contents
MAXDATERR	(none)	Specifies the maximum number of invalid dates permitted
MAXRECIN	MRI	Controls the number of records to input
MAXRECOUT	MRO	Controls the maximum number of records to output
MEMBER	M	Specifies a member name to process in a PDS
MEMBERS	MS	Specifies a group of members to process in a PDS with a mask
MOVE	MV	Moves data into the record
NEWMBR	NM	Gives a new name to an output PDS member
NEWMBRS	NMS	Names multiple new members of an output PDS using a mask
OPTIONS	OP	Controls STARBAT processing options
OR	(none)	Used with the IF parameter to indicate an OR condition
OVERALL	OA	Replaces all occurrences of data in a record with other data
OVERLAY	OL	Replaces the first instance of data in a record with new data
PADCHAR	PAD	Specifies a padding character for uninitialized parts of a record
PRINT	P	Prints records in alphanumeric format with record statistics
PRINTCHR	PC	Prints records in alphanumeric format
PRINTHEX	PH	Prints records in vertical hexadecimal format
PRINTLPI	PL	Specifies the number of lines per inch for print output pages
RBA	(none)	Processes VSAM data beginning at a relative byte address
RDW	(none)	Controls the inclusion of the record descriptor word
SELECT	S	Processes every nth record
STARTKEY	SK	Processes VSAM data beginning with a generic key
STOPIF	ST	Stops processing a function when a record satisfies a condition
SUM	(none)	Accumulates the contents of specified fields
WARP	(none)	Modifies data numeric values and performs date aging logic
WARPDEF	WD	Specifies default WARP values
WRITE	W	Writes a record to one or more output files

STARBAT Data Sets

STARBAT supports sequential, direct, VSAM, and partitioned data sets. ISAM data sets are not supported.

Sequential Data Sets

STARBAT can create any type of sequential data set as output. Input concatenated data sets on unlike devices are supported but these data sets must have similar characteristics. When processing concatenated input data sets, STARBAT treats all of these data sets as one logical input data set and no reports are provided on the individual data sets in the concatenation aside from a record count for each concatenated data set.

Direct Data Sets

STARBAT uses BSAM to read direct data sets. If RKP is greater than zero, you do not need to perform any special action to copy the data and the imbedded keys. However, if RKP is zero, you will need to use a MOVE parameter with a negative relative location to reference the key portion because the key is not also present in the data.

For example, if you have a record with a 20 byte key and a 80 byte data portion, use the following MOVE parameter to copy both the key and data portion of the record to an output record:

```
DD01  COPYREC   MOVE=(1,100,-20)
```

VSAM Data Sets

STARBAT supports RRDS, KSDS, and ESDS data sets. Linear VSAM data sets are supported but the CISIZE has to be 4K.

Partitioned Data Sets

STARBAT supports PDS and PDSEs. To process a specific member, use the MEMBER parameter. To process multiple members, either use multiple control statements with the MEMBER parameter (only one MEMBER parameter is allowed for each control statement), or use the MEMBERS parameter specifying a member mask. To process the entire data set, do not specify any MEMBER or MEMBERS parameter.

If the PDS and PDSEs are concatenated as input and if the MEMBER parameter is used, STARBAT processes each data set in the order of concatenation and stops processing the function as soon as the member is found.

There are a few restrictions when processing load library members:

1. Load PDSE members cannot be used as output in a copy.
2. The MULTICOPY function does not support scatter-loaded, overlay, or note-listed modules.
3. The block size of the input data set must be equal to or less than the output data set.
4. The main module associated with an alias member is always copied at the same time an alias is copied.

STARBAT JCL and DDNAMES

A sample of JCL to execute STARBAT follows.

```
//STARBAT1 JOB  (JOB CARD PARAMETERS)
//A   EXEC PGM=STARBAT,REGION=2048K
//STEPLIB  DD  DSN=STEPLIB.DATA.SET,DISP=SHR
//DD03     DD  DSN=SAMPLE.INPUT.DATA.SET,DISP=SHR
//DD03O    DD  DSN=SAMPLE.OUTPUT.DATA.SET,DISP=SHR
//SYSPRINT DD  SYSOUT=*
//SYSLIST  DD  SYSOUT=*
//SYSTOTAL DD  SYSOUT=*
//SYSIN    DD  *
DD03 COPYALL=(1,0,C'PGM=TESTPROG',C'PGM=PRODPROG'),
            PRINTCHR=100
```

The STARBAT job looks like any other MVS job. The usual rules for EXEC and DD statements apply.

The **EXEC** statement tells the operating system the name of the program to be executed. A two-megabyte region is usually sufficient for STARBAT.

A **STEPLIB** or **JOBLIB** statement is required unless STARBAT is present in your system's link list.

A **DDxx** statement is required to describe input data sets to STARBAT. The xx value can be any number from 00 through 99 that matches a control statement name like **DDxx**. Note that input data sets may be concatenated.

A **DDxxO** statement is required for **COPYREC** or **EXCLUDEREC** functions. The xx value must match the xx value in the input data set defined on the corresponding **DDxx** JCL statement. If necessary, basic DCB information such as RECFM, LRECL, and BLKSIZE will be copied from the input data set to the output data set when the data set is opened.

Additional output DD statements with any desired name may be specified for the **MULTICOPY** function; the WRITE parameter names the output DD statement and causes output records to be written when it is processed.

STARBAT primary output is directed to the **SYSPRINT** DD statement. This statement is optional; it will be dynamically allocated if it is not present. STARBAT echoes control statements and error, status, and completion messages to this DD name. STARBAT expects this to be a data set with standard attributes for a listing file namely: RECFM=FBA, LRECL=133, and BLKSIZE a multiple of 133. Note that an LRECL of 80 through 132 will work but will truncate some print lines.

If you specify the PRINT or PRINTHEX options, the **SYSLIST** DD statement should be included. If it is omitted, it is dynamically allocated if needed. This data set has the same DCB attributes as the SYSPRINT data set.

If you specify the SUM option, the **SYSTOTAL** DD statement should be included. If it is omitted, all SYSTOTAL output is re-directed to the SYSPRINT data set. This data set also has the same DCB attributes as the SYSPRINT data set.

The **SYSIN** statement provides control statements for STARBAT. Control cards must be in standard 80-character format and all 80 columns may be used. If no control cards are found, STARBAT defaults to a COPYREC function for every pair of matching input and output data sets (**`//DDxx`** and **`//DDxxO`**).

STARBAT Control Statements

STARBAT control statements consist of a data set identifier, a function name, and one or more parameters separated by commas. Comments are preceded by one or more blanks following a parameter.

In the following example, **DD01** is the data set identifier. **PRINTCHR** is the function name, **IF=(10,EQ,C'A')** is the parameter and the remainder of the statement is comments.

```
DD01 PRINTCHR IF=(10,EQ,C'A')  prints records containing character 'A' in column 10.
```

Data Set Identifier

The data set identifier identifies which data set in the JCL DD statement is being processed. The format of the data set identifier is DDxx where xx is from 00 to 99. The number references the //DDxx DD statement, which specifies the input data set to be processed. DDxx must begin in column 1 of the control statement. If the function is **COPY** or **EXCLUDEREC**, DDxx is also used to reference the //DDxxO DD statement representing the output data set.

Function Identifier

The function identifier identifies the action that you want to perform on the data. It is always positioned after the data set identifier with one or more spaces.

Parameters

Parameters are conditions defining the scope of the function; they are positioned after the function identifier and one or more spaces. Each parameter is followed by an equal sign and one or more elements containing a combination of start, length, operator, and data fields. There can be one or more parameters separated by commas.

Start

A start location indicates the location where data can be found in the input record. A start location may be coded in one of two ways: actual start location and relative start location.

Actual start location is the exact column number where the data in the record is positioned. The number can be from 1 to 32767, but it cannot be larger than the record size.

Relative start location indicates the position relative to the current location. STARBAT supports both input relative start location and output relative start location.

When an input record is first read by STARBAT, the relative location is set to the beginning of the record. Parameters such as IF, CHANGE, and OVERLAY change this location value. A relative start location can be specified by placing a plus sign or a minus sign before a number. For example, to reference a location 10 bytes before the current location, use –10; to reference a location 5 bytes after the current location, use +5. Output relative location is supported when using the MOVE parameter with the COPY or MULTICOPY functions. When data is moved to an output location, the output relative location moves to the next available output position.

Operator

Operators are EQ, NE, GT, LT, GE, and LE. They are used to compare data with an input record.

STARBAT also supports bit value comparisons using operators AO (all ones), AZ (all zeroes), NO (not ones), and MX (mixed ones and zeroes) with the IF, OR, CHANGE, or OVERLAY parameters.

Length

When the exact location of the compare data is unknown, the length element may be used in place of the operator element. In this case, an equal comparison is assumed.

To scan the entire input record, use a length of 0. Otherwise, the length value must be at least one greater than the length of the compare data; also, the sum of the current input location and the length value must be less than or equal to the record length. Length cannot be greater than 255.

Note that the IF, OR, CHANGE and OVERLAY parameters are changed to scanning parameters when the length element is used. Therefore, they will change the relative start input position as previously mentioned.

Data

STARBAT supports character, hexadecimal, binary, and packed data. The data element can be used as compare data, as replacement data, or as literal data.

Character data may be enclosed in single or double quotes. When used as compare data, more than one value can be entered with commas in between, indicating an OR condition, as long as they are enclosed in single quotes. For example, IF=(10,EQ, C'TEST,PROD') checks if the record contains the characters TEST or PROD in column 10. A duplication factor may be used to avoid coding repetitive data elements. For example, instead of coding C'TEST,PROD,TEST,PROD,', you could code 2C'TEST,PROD,'.

Alphanumeric data is case-sensitive and is not translated to upper case. To search for data in both upper and lower case, use the T character string. For example, to search for PROD, Prod, or proD, use T'Prod'.

Note that all character data must fit into a single control statement because continuations are not supported.

Packed data can be entered with any valid length that fits on a single control statement (continuations are not supported). You must enclose all numeric digits in single quotes. STARBAT calculates the data length using the data entered. You may enter a plus or minus before each string of numbers; an unsigned packed number is considered positive. For example, `P'12'` and `P'+12'` are equivalent.

Binary data can be entered with a beginning H character for halfword binary numbers or an F character for fullword binary numbers. You must enclose all numeric digits in single quotes. You may enter a plus or minus before each string of numbers; an unsigned binary number is considered positive. For example, `F'12'` and `H'-12'`.

Hexadecimal data can be entered with any length that fits on a single control statement (continuations are not supported). You must enclose all hexadecimal digits in single quotes. STARBAT calculates the data length using the data entered and it must always have an even number of digits. You may also use a numeric duplication factor to avoid coding repetitive data elements. For example, `3X'003F'` and `X'003F003F003F'` are equivalent.

Comments

Comments may be coded after a parameter or a parameter with a comma as long as there are one or more blanks between them. A comment may also be coded as a standalone statement provided that the statement begins with an *. In this case, the comment will also be output in the SYSTOTAL data set.

Three comments are coded in the example below:

```
DD03 COPYREC MAXRECIN=10, COMMENT#1
             COPYOVER=YES COMMENT#2
* COMMENT#3
```

STARBAT Functions

Following is a brief description of each supported function.

Function	Short	Description
COPYALL	CA	Copies all records of a data set
COPYMBR	CM	Copies members conditionally based on contents
COPYREC	CR	Copies data and reports records copied
COPYREV	CPR	Copies data in reverse order
COPYSOME	CS	Copies selected records but applies all changes like COPYALL
EXCLUDEREC	XR	Eliminates unwanted records in a copy
MULTICOPY	MC	Copies data to one or more output data sets
PRINT	P	Prints records in alphanumeric format with record statistics
PRINTALL	PA	Prints all records of a data set
PRINTCHR	PC	Prints records in alphanumeric format
PRINTCHRALL	PCA	Prints all records of a data set
PRINTCHRMBR	PCM	Prints members conditionally based on contents
PRINTCHRREV	PCR	Prints records in alphanumeric format and reverse order
PRINTHEX	PH	Prints records in vertical hexadecimal format
PRINTHEXALL	PHA	Prints all records of a data set
PRINTHEXMBR	PHM	Prints members conditionally based on contents
PRINTHEXREV	PHR	Prints records in vertical hexadecimal format and reverse order
PRINTMBR	PM	Prints members conditionally based on contents
PRINTREV	PRR	Prints records in alphanumeric format and reverse order
SKIP	S	Moves the current record pointer forward
SKIPREV	SKR	Moves the current record pointer backward
TOTAL	T	Reads input records processing all parameter groups for SUM
UPDATEALL	UA	Updates all records of a data set
UPDATEMBR	UM	Updates members conditionally based on contents
UPDATEREC	UR	Updates records in place

STARBAT function names can be abbreviated as shown in the title lines in the following pages.

COPYREC or CR, COPYALL or CA, COPYMBR or CM, COPYSOME or CS Function

Copies data and reports records output. If it is required to copy records from one data set into another, the **COPYREC** function is used. With the addition of qualifiers it is possible to control the records selected for copying. Additionally, in the process of copying, the data can be modified by expanding or contracting fields and changing the data within fields.

Use **COPYMBR** to copy members based on their contents and use **COPYALL** to process multiple conditional updates (IF OVERLAY, IF CHANGE, IF MOVE, IF EXPAND, IF WARP) while copying all records. Use the **COPYSOME** function to process multiple conditional updates like **COPYALL** while copying selected records like **COPYREC**.

The following example will copy up to 1000 records that contain "USA" anywhere after position 256:

```
DD01 COPYREC IF=(256,0,C'USA'),MAXRECOUT=1000
```

The following example will copy the input data set and replace the first "USA" in the first three positions with "DOM":

```
DD01 COPYREC OVERLAY=(1,3,C'USA',C'DOM')
```

The following example will copy every other record that contains "USA" in the first three positions:

```
DD01 COPYREC SELECT=2,IF=(1,3,C'USA')
```

The following example will copy all members whose names begin with STAR containing "USA" anywhere in the member:

```
DD01 COPYMBR MEMBERS=STAR-,IF=(1,0,C'USA')
```

The following example will copy records containing an 8 or 9 and warp each record with YYDDD or YYMMDD format. Note that if a record has both formats, only the YYDDD format will be warped.

```
DD01 COPYREC   IF=(12,EQ,C'8'),
               WARP=(16,C,YYDDD,ADD=5Y),
               IF=(22,EQ,C'9),
               WARP=(26,C,YYMMDD,ADD=5Y)
```

The following example will copy all records and warp all 8 and 9 dates:

```
DD01 COPYALL   IF=(12,EQ,C'8'),
               WARP=(16,C,YYDDD,ADD=5Y),
               IF=(22,EQ,C'9),
               WARP=(26,C,YYMMDD,ADD=5Y)
```

The following example will copy records containing an 8 or 9 and warp all dates on these records:

```
DD01 COPYSOME  IF=(12,EQ,C'8'),
               WARP=(16,C,YYDDD,ADD=5Y),
               IF=(22,EQ,C'9),
               WARP=(26,C,YYMMDD,ADD=5Y)
```

COPYREV or CPR Function

Copies records from a sequential or VSAM data set in reverse order. This means that the first record copied will be the last record from the input data set followed by other records from the input data set moving toward the front of the data set.

The following example will copy records except those containing RED anywhere in reverse order:

```
DD01 COPYREV IF=(1,0,C'RED')
```

The following example will copy up to 1000 records that contain "USA" anywhere after position 256 in reverse order:

```
DD01 COPYREV IF=(256,0,C'USA'),MAXRECOUT=1000
```

EXCLUDEREC or XR Function

Excludes (or eliminates) records during a copy; **EXCLUDEREC** selects records that would be bypassed by **COPYREC**.

The following example will copy all records except those containing RED anywhere:

```
DD01 EXCLUDEREC IF=(1,0,C'RED')
```

The following example will copy all records except the first 100 containing RED anywhere:

```
DD01 EXCLUDEREC IF=(1,0,C'RED'),EXCLUDEREC=100
```

The following example will copy the input data set and drop any record that contains an A in position 1 or contains a value higher than CAA in location 112. Processing stops after copying 20 records:

```
DD01 EXCLUDEREC IF=(1,EQ,C'A'),OR=(112,GT,C'AA'),MAXRECOUT=20
```

MULTICOPY or MC Functions

Copies data to one or more output data sets. With the **MULTICOPY** function, you can direct output records to several output data sets and add, delete, or modify records being copied to the output data sets at the same time.

In order to control the destination of the records, the WRITE parameter is used to specify the output DDNAME.

Output records for **MULTICOPY** are created piece by piece with MOVE or by copying the input record. If the MOVE parameter is used, the output area is initialized to the PADCHAR value and not reset between different WRITE parameters.

There are several restrictions on the data that can be output with a **MULTICOPY** function:

1. Output to load members is supported; however, JCL indicating PDS(MEMBER) notation is not supported. Instead, use only the PDS name in the JCL and indicate the member name with the MEMBER, MEMBERS, NEWMBR, or NEWMBRS parameters.
2. Overlay, scatter-loaded, and note-listed load members are not supported.

The following example shows how to create two identical output data sets in **//OUT1** and **//OUT2**:

```
DD01 MULTICOPY WRITE=(OUT1,OUT2)
```

The following example will copy a file, repeat any record with a C'5' and change it to a C'6':

```
DD01 MULTICOPY WRITE=OUT1,IF=(20,EQ,C'5'),MOVE=(1,0,1),
            MOVE=(20,C'6'),WRITE=OUT1
```

The following example will add a STEPLIB JCL statement after any EXEC statement:

```
DD01 MULTICOPY WRITE=NEWJCL,IF=(1,20,C' EXEC '),MOVE=(1,80C' '),
            MOVE=(1,C'//STEPLIB  DD  DISP=SHR,'),
            MOVE=(+0,C'DSN=MYHILEV.MYMIDLEV.MYLOWLEV'),WRITE=NEWJCL
```

PRINT or P, PRINTALL or PA, PRINTMBR or PM, PRINTREV or PRR Functions

Prints a data set or a portion of a data set in alphanumeric format with identifying information such as record number, RBA, and record length. A column scale is normally printed with each record; however, if you request OPTIONS=SHORT, the column scale will be produced only after each page header.

Use **PRINTMBR** to print members based on their contents; use **PRINTALL** to process multiple conditional updates (IF OVERLAY, IF CHANGE, IF MOVE, IF WARP) while printing records; use **PRINTREV** to print sequential or VSAM data sets in reverse order.

The following example will print the first 10 records:

```
DD01 PRINT MAXRECOUT=10
```

The following example will print the first 20 records containing RED in position 12 replaced with BLU. Note that the input data set is not actually changed; use this function to see changes before running an **UPDATEREC**:

```
DD01 PRINT OVERLAY=(12,EQ,C'RED',C'BLU'),MAXRECOUT=20
```

The following example will print the first 100 positions of the first 20 records that contain RED in position 12:

```
DD01 PRINT IF=(12,EQ,C'RED'),MAXRECOUT=20,MOVE=(1,100,1)
```

The following example will print the first 100 positions of the last 20 records with RED in position 12 in reverse order:

```
DD01 PRINTREV IF=(12,EQ,C'RED'),MAXRECOUT=20,MOVE=(1,100,1)
```

PRINTCHR or PC, PRINTCHRALL or PCA, PRINTCHRMBR or PCM, PRINTCHRREV or PCR Functions

Prints a data set or a portion of a data set in alphanumeric format with no additional record information. If you wish to print a ruler at the top of each page to help identify the columns used, you can specify the OPTIONS=SHORT parameter.

Use **PRINTCHRMBR** to print members based on their contents; use **PRINTCHRALL** to process multiple conditional updates (IF OVERLAY, IF CHANGE, IF EXPAND, IF MOVE, IF WARP) while printing records and use **PRINTCHRREV** to print sequential or VSAM data sets in reverse order.

The following example will list the first 10 records:

```
DD01 PRINTCHR MAXRECOUT=10
```

The following example will list the first 20 records containing RED in position 12 replaced with BLU. Note that the input data set is not actually changed; use this function to see changes before running an **UPDATEREC**:

```
DD01 PRINTCHR OVERLAY=(12,EQ,C'RED',C'BLU'),MAXRECOUT=20
```

The following example will list the first 100 positions of the first 20 records that contain RED in position 12:

```
DD01 PRINTCHR IF=(12,EQ,C'RED'),MAXRECOUT=20,MOVE=(1,100,1)
```

The following example will list the last 10 records in reverse order:

```
DD01 PRINTCHRREV MAXRECOUT=10
```

PRINTHEX or PH, PRINTHEXALL or PHA, PRINTHEXMBR or PHM, PRINTHEXREV or PHR Functions

Prints records in a vertical hexadecimal format. The report produced also supplies additional useful information such as the record number and the record length. A column scale is normally printed with each record; however, if you request OPTIONS=SHORT, the column scale will only be produced after each page header.

Use **PRINTHEXMBR** to print members based on their contents; use **PRINTHEXALL** to process multiple conditional updates (IF OVERLAY, IF CHANGE, IF EXPAND, IF MOVE, IF WARP) while printing records and use **PRINTHEXREV** to print sequential or VSAM data sets in reverse order.

To print the first 10 records of a file in vertical hexadecimal:

```
DD01 PRINTHEX MAXRECOUT=10
```

The following example will print, in vertical hexadecimal, the first 20 records containing RED in position 12 replaced with BLU. Note: the input data set is not actually changed; use this function to see changes before using the update functions.

```
DD01 PRINTHEX OVERLAY=(12,EQ,C'RED',C'BLU'),MAXRECOUT=20
```

This example will print, in vertical hexadecimal, up to 100 positions of the first 20 records with RED in position 12:

```
DD01 PRINTHEX IF=(12,EQ,C'RED'),MAXRECOUT=20,MOVE=(1,100,1)
```

The following example will print, in vertical hexadecimal, the last 10 records in reverse order:

```
DD01 PRINTHEXREV MAXRECOUT=10
```

SKIP or S, SKIPREV or SKR Function

Moves the current record pointer forward or backward. This allows large groups of records to be excluded from processing.

Note that you may not follow a **SKIP** function with an **UPDATEREC** function because the data set must be closed and reopened while maintaining data set positioning. Note that you also cannot follow a **SKIP** function with a reverse function like **COPYREV** and you can not follow a **SKIPREV** function with a forward operation like **COPYREC**.

The following example will skip the first 100 records before a **PRINT** function:

```
DD01 SKIP MAXRECIN=100
DD01 PRINT IF=(10,EQ,C'RED')
```

The following example will skip the first 100 records before an **UPDATEREC** function:

```
DD01 UPDATEREC  MAXRECIN=100
DD01 UPDATEREC OVERLAY=(10,EQ,C'RED',C'BLU')
```

The following example will skip the last 100 records before a **PRINTREV** function:

```
DD01 SKIPREV MAXRECIN=100
DD01 PRINTREV IF=(10,EQ,C'RED')
```

TOTAL or T Functions

Reads input records processing all parameter groups for SUM. The **TOTAL** function creates simple reports that can be used to validate data in a data set. The results generated from the **TOTAL** function and the SUM parameter are directed to the SYSTOTAL DDNAME if it is present; otherwise, the results are directed to the SYSPRINT data set. Comment statements can also be added to the input control statements and they will be printed with the **TOTAL** function results.

STARBAT will produce a two line data set identification message after the comment and before the accumulation with the following output identifier: "**FOLLOWING TOTALS DEVELOPED FROM**" for the first line and the input data set name and volume name on the second line.

The following example will accumulate all packed numbers beginning in column 43:

```
DD01 TOTAL  SUM=(43,'This is the packed data set total')
```

The following example will accumulate all character numbers beginning in column 43 for type M records:

```
DD01 TOTAL  IF=(22,EQ,C'M'),SUM=(43,6,C,'This is the character data set total')
```

UPDATEREC or UR, UPDATEMBR or UM, UPDATEALL or UA Functions

Updates records in place. This function must be combined with any of the parameters used to modify data such as the CHANGE and OVERLAY parameters.

Note that this will update the data set in place. If you wish to preview your changes prior to committing them, use the **PRINT** functions instead to show the changes that will be made and then re-run the job replacing the **PRINT** function with the **UPDATEREC** function.

Use **UPDATEMBR** to update members based on their contents and use **UPDATEALL** to process multiple conditional updates (IF OVERLAY, IF CHANGE or IF WARP) while processing all records.

The following example will show potential changes before performing an actual **UPDATEREC**:

```
DD01 PRINT  OVERLAY=(1,0,C'RED',C'BLU')
```

The following example will change the first occurrence of RED anywhere in a record to BLU:

```
DD01 UPDATEREC OVERLAY=(1,0,C'RED',C'BLU')
```

The following example will change all occurrences of RED to BLU and all occurrences of WHI to ANY:

```
DD01 UPDATEALL OVERALL=(1,0,C'RED',C'BLU'),
                 OVERALL=(1,0,C'WHI',C'ANY')
```

STARBAT Parameters

Parameters may be specified on function statements to limit the data processed and to control the function.

In this section, the following parameters are described in alphabetical order.

Parameter	Short	Description
ABEND	AB	Controls EOJ processing when an abnormal condition occurs
AND	IF	Creates a logical AND condition check (used with IF)
CHANGE	C	Changes only the first instance of data in a record
CHANGEALL	CA	Changes all occurrences of data in a record
COPYOVER	CO	Controls the replacement of identically named output members
EXCLUDEREC	XR	Controls the # of records to bypass in an EXCLUDEREC function
EXPAND	EX	Expands records at a specified location
IF	AND	Selects records to process based on data contents
MAXDATERR	(none)	Specifies the maximum number of invalid dates permitted
MAXRECIN	MRI	Controls the number of records to input
MAXRECOUT	MRO	Controls the maximum number of records to output
MEMBER	M	Specifies a member name to process in a PDS
MEMBERS	MS	Specifies a group of members to process in a PDS with a mask
MOVE	MV	Moves data into the record
NEWMBR	NM	Gives a new name to an output PDS member
NEWMBRS	NMS	Names multiple new members of an output PDS using a mask
OPTIONS	OP	Controls STARBAT processing options
OR	(none)	Used with the IF parameter to indicate an OR condition
OVERALL	OA	Replaces all occurrences of data in a record with other data
OVERLAY	OL	Replaces the first instance of data in a record with new data
PADCHAR	PAD	Specifies a padding character for uninitialized parts of a record
PRINT	P	Prints records in alphanumeric format with record statistics
PRINTCHR	PC	Prints records in alphanumeric format
PRINTHEX	PH	Prints records in vertical hexadecimal format
PRINTLPI	PL	Specifies the number of lines per inch for print output pages
RBA	(none)	Processes VSAM data beginning at a relative byte address
RDW	(none)	Controls the inclusion of the record descriptor word
SELECT	S	Processes every nth record
STARTKEY	SK	Processes VSAM data beginning with a generic key
STOPIF	ST	Stops processing a function when a record satisfies a condition
SUM	(none)	Accumulates the contents of specified fields
WARP	(none)	Modifies data numeric values and performs date aging logic
WARPDEF	WD	Specifies default WARP values
WRITE	W	Writes a record to one or more output files

Parameters are grouped according to type as follows:

Action	Changes data (CHANGE, EXPAND, MOVE, OVERLAY, OVERALL, SUM, WARP and WRITE)
Limit	Specifies record count limits (EXCLUDEREC, MAXRECIN, MAXRECOUT and SELECT)
Print	Prints records as they are processed (PRINT, PRINTCHR and PRINTHEX)..
Selection	Selects records based on their contents (AND, IF, and OR)
Control	Defines basic conditions during execution (all other parameters such as ABEND and STOPIF)

Several restrictions on parameter and function combinations should be noted:

1. The EXCLUDEREC parameter can only be used with the **EXCLUDEREC** function.
2. The WRITE parameter can only be used with the **MULTICOPY** function.
3. The PRINTHEX, PRINTCHR and PRINT parameters should not be used with the **PRINT** functions.
4. The EXPAND, MOVE and MAXRECOUT parameters can not be used with the **UPDATE** functions.
5. The NEWMBR, NEWMBRS and COPYOVER parameters can not be used with **PRINT** functions, **SKIP, SKIPREV** or **UPDATEREC**.

Many STARBAT parameters process numeric character data and packed decimal data. Numeric characters must be in zoned decimal (hexadecimal X'F0' through X'F9') and the last byte can be signed positive (X'C0'), negative (X'D0') or unsigned (X'F0'). Packed decimal numbers must contain valid numeric numbers and the sign must be positive (with either X'0C' or X'0F') or negative (X'0D'). In most cases, you can enter a data length of 0 for packed decimal numbers and STARBAT will determine the length dynamically by scanning for the sign digit in each field as it is processing each record.

ABEND or AB Parameter

The ABEND parameter controls how the end of job processing should be handled when an abnormal condition occurs during job execution.

```
ABEND=0/1/2
```

ABEND=0 Issues a return code at normal end of job.
ABEND=1 Issues a U0012 abend dump when an I/O error occurs. This is the default.
ABEND=2 Issues a user abend when a non-zero return code is encountered.

In the following example, a user abend of U0008 will occur if the character string ‘ALIAS-‘ is not found in the input data set and no records are written to the output data set.

```
DD01 MULTICOPY ABEND=2,IF=(1,0,C'ALIAS-').MOVE=(19,0,80),WRITE=FILE1
```

AND or IF Parameter

The AND parameter selects records to be processed by the function being executed. The AND parameter and the IF parameter have identical meanings and format. AND is normally used after an IF parameter to improve readability.

For additional details, see the IF parameter.

CHANGE or C Parameter

The CHANGE parameter works like the CHANGE command in ISPF edit. It replaces the value in string-1 with the value in string-2. String-1 and string-2 can be different lengths and the data will be shifted left or right depending on the size of the second string (see NOTE: below). CHANGE will only modify the first occurrence of the matching data in the record, as with ISPF. If you wish to change multiple occurrences, use the similar CHANGEALL parameter.

```
CHANGE=(start,length/operator,string-1,string-2)
```

start defines the position of the record where the search is to begin. The first byte of the record is position 1.
length specifies how many characters to search. Specify 0 if you want to search the remainder of the record.
operator an alternative to length is to specify a relational operator (such as EQ).
string-1 is a binary, character, hexadecimal or packed string representing data to be searched for at this location.
string-2 is a binary, character, hexadecimal or packed string representing data to be replaced at this location.

NOTE: When CHANGE replaces a data field by a shorter field, STARBAT shifts contiguous characters to the left until a blank character is found and inserts blanks to adjust for the missing characters. If no blanks are found, blanks are inserted at the end of the record.

For example, with `CHANGE=(1,5,C'1234',C'X')`

```
Record 1: 12345 AB1234GH
Record 2: 123456ABCDEFGH
     would become:
Record 1: X5    AB1234GH
Record 2: X56ABCDEFGH
```

When CHANGE replaces a data field by a longer field, STARBAT shifts contiguous characters to the right compressing multiple blanks to single blanks until all characters fit into the record. Note: if no blanks are found using the **COPYREC** or **MULTICOPY** function for fixed length output records or using the **UPDATE** functions, truncation will occur if the data extends beyond the record boundary.

For example, with `CHANGE=(11,3,C'USA',C'DOMESTIC')`

```
Record 1: AA19980101USA       CA94010-1904
Record 2: AA19880101USA  CA94010-1904
     would become:
Record 1: AA19980101DOMESTIC  CA94010-1904
Record 2: AA19880101DOMESTIC CA94010-1904
```

It is important to remember when using the **CHANGE** (or **OVERLAY**) parameters that they operate on the input data records. Therefore, you should specify any MOVE functions after the **CHANGE** (or **OVERLAY**) parameter because **MOVE** operates on the output record.

CHANGEALL or CA Parameter

The CHANGEALL parameter is the same as the CHANGE parameter except that all occurrences are modified and the second parameter must be a length value (not an operator like EQ). See CHANGE for additional details.

```
CHANGEALL=(start,length,string-1,string-2)
```

start defines the position of the record where the search is to begin. The first byte of the record is position 1.
length specifies how many characters to search. Specify 0 if you want to search the remainder of the record.
string-1 is a binary, character, hexadecimal or packed string representing data to be searched for at this location.
string-2 is a binary, character, hexadecimal or packed string representing data to be replaced at this location.

COPYOVER or CO Parameter

The COPYOVER parameter may be used to specify if identically named output members are to be replaced when copying from one PDS to another.

```
COPYOVER=YES/NO
```

COPYOVER=YES Default, replaces the output PDS member if it already exists.
COPYOVER=NO Do not replace the output PDS member if it already exists.

The following example specifies that existing members are not to be replaced:

```
DD01 COPYREC COPYOVER=NO
```

EXCLUDEREC or XR Parameter

The EXCLUDEREC parameter is used with the **EXCLUDEREC** function to control the number of records to exclude.

```
EXCLUDEREC=n
```

n is a number from 1 to 999999999. It represents the number of records to be excluded.

The following example replaces all occurrences of 'MAIL' with 'IMA1' and omits the first record that contains 'IMA1-M'.

```
DD01 EXCLUDEREC OVERALL=(1,0,C'MAIL',C'IMA1'),IF=(1,EQ,C'IMA1-M'),EXCLUDEREC=1
```

EXPAND or EX Parameter

The EXPAND parameter is used to expand date and data fields. EXPAND is similar to MOVE but the original record is preserved to the extent possible. Character type is normally used for date expansions of any format because after a date is padded on the right with blanks; STARBAT can still read it using its input picture and rewrite it with a new picture.

Column numbers referred to by EXPAND are the original column numbers in the data; similarly, references to column numbers by other STARBAT parameters should also reference the original column numbers because STARBAT will adjust these column numbers dynamically.

```
EXPAND=(column,type,size,newsize)
```

column is the location where the data item begins. Any valid actual or relative location may be used.
type is **B** for binary, **P** for packed, **C** for character or **N** for character numeric. Character type pads the original item on the right with blanks; other types of field expansion add numeric padding to the left.
size is the number of bytes used by the original data item.
newsize is the number of bytes to be used by this data item. **Newsize** can be smaller or larger than **size**.

The following example will expand a packed data field and accumulate the next character field:

```
DD01 COPYREC  EXPAND=(21,P,4,8),SUM=(25,5,C,'The next field')
```

The following example will expand a date field and convert a character date format from YYDDD to CCYYDDD:

```
DD01 COPYREC  EXPAND=(21,C,5,7),WARP=(21,C,YYDDD,OUTPIC=CCYYDDD)
```

The following example will expand a date field and convert a packed date format from YYDDD to CCYYDDD:

```
DD01 COPYREC  EXPAND=(21,C,3,4),WARP=(21,P,YYDDD,OUTPIC=CCYYDDD)
```

The following example will expand a binary field and a numeric character field and accumulate the fields:

```
DD01 COPYREC  EXPAND=(12,B,2,4),EXPAND=(21,N,4,9),SUM=(12,4,B),SUM=(21,9,C)
```

IF or AND Parameter

The IF parameter selects records to be processed by the function being executed. The IF parameter and the AND parameter have identical meanings and format. AND is normally used after an IF parameter to improve readability.

There are two IF syntax forms to test for data contents or valid numerics:

```
   IF=(start,length/operator,string,...)     /* data content test   */
   IF=(start,length,[duplicate]type,...)     /* valid numeric test  */
```

start	defines the position of the record where the search is to begin. The first byte of the record is position 1.
length	specifies how many characters to search. Specify 0 if you want to search the remainder of the record.
operator	an alternative to length is to specify a relational operator (such as EQ).
string	is a binary, character, hexadecimal or packed string representing data to be searched for at this location.
[duplicate]type	is an optional duplication number followed by **EQN** and **EQP** to check for valid numeric character or packed decimal or **NEN** and **NEP** to check for invalid numeric character or packed decimal.

Note: you may enter multiple parameter sets in an IF parameter separated by commas to obtain multiple logical OR tests.

In the following example, all records without a hex 'F0F1' in column 1 are written to file FILE1.

```
DD01 MULTICOPY IF=(1,NE,X'F0F1'),WRITE=FILE1
```

The following example copies all records, but in those records having an 'ME' in position 1 AND a 'QE' in column 3, position 10 will be replaced with a 'P'.

```
DD01 COPYALL IF=(1,EQ,T'ME'),IF=(3,EQ,C'QE'),OVERLAY=(10,C'P')
```

The example above can also be coded using the AND parameter.

```
DD01 COPYALL IF=(1,EQ,T'ME'),AND=(3,EQ,C'QE'),OVERLAY=(10,C'P')
```

The following example directly updates the input data set. If the data at position 60 for a length of 5 is not numeric, position 60 will be replaced with five zeros.

```
DD01 UPDATEREC IF=(60,5,NEN),OVERLAY=(60,C'00000')
```

In the following example, all records with invalid packed numbers will be copied to the output data set:

```
DD01 COPYREC IF=(44,0,NEP),PRINTHEX=4
```

This example will copy input records with a 1 in position 10 and a 2 in position 20 or an A in position 3 and a B in position 6. Since it is a COPYREC, only position 30 would be warped; COPYSOME or COPYALL would warp both positions.

```
DD01 COPYREC IF=(10,EQ,C'1'),AND=(20,EQ,C'2'),WARP=(30,C,CCYYMMDD,ADD=5Y),PRINT=3,
            IF=(3,EQ,C'A'),AND=(6,EQ,C'B'),WARP=(40,C,CCYYMMDD,ADD=10D),MAXRECOUT=30
```

The following example will print, in vertical hex, all records that contain odd EBCDIC numbers in column 14:

```
DD01 PRINTHEX IF=(14,AO,X'F1')
```

The following example will print, in vertical hex, all records that contain even EBCDIC numbers in column 14:

```
DD01 PRINTHEX IF=(14,GE,X'F0'),IF=(14,NO,X'01')
```

The following example will print, in vertical hex, all records that contain an EBCDIC 0, 4 or 8 in column 14:

```
DD01 PRINTHEX IF=(14,AO,X'F0'),IF=(14,AZ,X'03')
```

MAXDATERR Parameter

The MAXDATERR parameter controls the number of invalid date fields or overflow data errors to permit from WARP parameters before terminating STARBAT.

```
MAXDATERR=n
```

n defaults to 100 and specifies the maximum number of invalid date fields or overflow errors to permit.

The following example will permit 3 invalid dates before terminating STARBAT.

```
DD01 COPYREC MAXDATERR=3,WARP=(12,C,YYDDD,ADD=12Y)
```

MAXRECIN or MRI Parameter

The MAXRECIN parameter specifies the number of input records to be read before terminating function processing.

```
MAXRECIN=n
```

n is the maximum records to be input. Any number from 0 to 999999999 can be used where 0 is all records.

The following example will copy the first 100 records from the input data set to the output data set:

```
DD01 COPYREC MAXRECIN=100
```

MAXRECOUT or MRO Parameter

The MAXRECOUT parameter controls the number or records to be printed or written before processing stops. MAXRECOUT can be used to limit the number of output records or extend the number of printed records past the default of 250. Note that the MAXRECOUT parameter is ignored for the update functions.

```
MAXRECOUT=n
```

n is the maximum records to output. Any number from 0 through 999999999 may be used where 0 is all records.

The following example will print, in vertical hex, the first 30 records that contain the string TEST RECORD in column 23:

```
DD01 PRINTHEX IF=(23,EQ,C'TEST RECORD'),MAXRECOUT=30
```

MEMBER or M Parameter

Specifies a member name to process in a PDS. If neither MEMBER nor MEMBERS is specified for a PDS, all members of the data set will be processed.

```
MEMBER=memname/(member1,member2,…)
```

memname specifies the member to be processed. To specify multiple members, use parentheses with MEMBER; use the MEMBERS parameter; or use multiple control statements with MEMBER and the same **DDxx** name.

The following example will copy a single member.

```
DD01 COPYREC MEMBER=MEMBERA
```

The following example will print three different members in vertical hex.

```
DD01 PRINTHEX MEMBER=MEMBERA
DD01 PRINTHEX MEMBER=MEMBERB
DD01 PRINTHEX MEMBER=MEMBERC
```

The following example will print three different members in vertical hex.

```
DD01 PRINTHEX MEMBER=(MEMBERA,MEMBERB,MEMBERC)
```

MEMBERS or MS Parameter

Specifies a group of members to process in a PDS with a mask. If neither MEMBER nor MEMBERS is specified for a PDS, all members of the data set will be processed.

```
   MEMBERS=ALL/maskname
```

ALL specifies that all members are to be selected.

maskname specifies the members to be processed using a mask of up to eight characters which is matched with member names from the data set. Several mask examples are shown below; for more information, see the Member Name Forms Appendix in the STARWARP Reference manual.

The following example will copy all members whose names begin with STAR.

```
DD01 COPYREC MEMBERS=STAR
```

Dash: Copy all members whose names begin with STAR and contain 98 in positions seven and eight.

```
DD01 COPYREC MEMBERS=STAR--98
```

Combination: Copy all members whose names begin with STAR and contain 98 elsewhere.

```
DD01 COPYREC MEMBERS=STAR*98
```

Range: Copy all members in the alphabetic range whose names begin with ABC through DEF999999.

```
DD01 COPYREC MEMBERS=ABC:DEF
```

Pattern: Copy all members whose names contain STAR and 98 anywhere.

```
DD01 COPYREC MEMBERS=STAR/98
```

MOVE or MV Parameter

The MOVE parameter builds an output record by moving data to it. Data from control statements or from input records can be used depending on which form of the MOVE parameter is used. Note: MOVE can not be used with the update functions. There are two MOVE syntax forms that move data from a control statement or from an input record:

```
  MOVE=(toposition,data)                Comment: data from control statement
  MOVE=(toposition,length,from-loc)     Comment: data from input record
```

toposition defines the starting position in the output record to which the data is to be moved. Absolute and relative positions can be used. 'n' means at position 'n', '+n' means 'n' bytes further down the record, '-n' means 'n' bytes previous in the record and '+0' means in the next available position.

data is a binary, character, hexadecimal or packed string representing data to be moved to the output record.

length defines the number of bytes to be moved. The number can be in the range 1 to 255. A value of 0 indicates that the remainder of the input record is to be moved.

fromposition defines the starting position in the input record from which the data is to be moved. Absolute and relative positions can be used as for toposition; '+0' references the input relative location or the last scan position.

The output record area is initially set to X'00' by default but this can be overridden by the PADCHAR parameter. A data value set in the output record remains set until new data is placed there; the output record does not need to be reinitialized.

The following example will move REDRED to the first six positions of an output record.

```
DD01 COPYREC MOVE=(1,2C'RED')
```

The following example will print the first 100 characters of each input record.

```
DD01 PRINT MOVE=(+0,100,+0)
```

The following example will substitute RED for the first three positions of each record.

```
DD01 COPYREC MOVE=(1,C'RED'),MOVE=(+0,0,4)
```

NEWMBR or NM Parameter

Names a new member of an output PDS.

```
   NEWMBR=memname
```

memname specifies a new name to be assigned to the copied member.

The following example will copy MEMBERA to an output data set and rename it to MEMBERCC.

```
DD01 COPYREC MEMBER=MEMBERA,NEWMBR=MEMBERCC
```

Note that an existing member will be replaced unless the COPYOVER=NO parameter is specified.

```
DD01 COPYREC MEMBER=MEMBERA,NEWMBR=MEMBERCC,COPYOVER=NO
```

NEWMBRS or NMS Parameter

Assigns new names to a group of output PDS members .using a mask.

```
   NEWMBRS=maskname
```

maskname The maskname can be from 1 to 8 characters long. Characters in the member name are replaced by characters from the maskname unless the character in the maskname contains a minus sign; the character in that position will be retained. Any blanks in the resulting member names are discarded.

The following example selects all members prefixed with TST and renames them to SYS..A. For example, member TST3333 will get renamed to SYS33A3. Members that contain the same name in the output data set will not be replaced.

```
DD01 COPYMBR MEMBERS=TST,NEWMBRS=SYS--A,COPYOVER=NO
```

This example will copy those members prefixed with TST and containing 'ALPHA' in column 1 in the data, and renames the selected members to PARM…

```
DD01 COPYMBR MEMBERS=TST,IF=(1,EQ,C'ALPHA'),NEWMBRS=PARM
```

OPTIONS or OP Parameter

Controls STARBAT processing options.

```
   OPTIONS=LONG/SHORT/JCL/MULTI/NOMULTI
```

OPTIONS=LONG default, a column scale is to be printed for PRINT and PRINTHEX after each data record.
OPTIONS=SHORT a column scale is only to be printed at the top of each PRINT, PRINTCHR and PRINTHEX page.
OPTIONS=JCL logical JCL processing is desired for DD, EXEC, JOB, PROC and SET JCL statements.
OPTIONS=MULTI after an end of file condition, a function with the same DDNAME is to reread the data set.
OPTIONS=NOMULTI default; after an end of file condition, another read will note another end of data set condition.

The following example will print a column scale only on each page.

```
DD01 COPYREC OPTIONS=SHORT,MEMBER=MEMBERA,PRINTHEX=500
```

The following example will utilize logical JCL processing for this control statement group.

```
DD01 COPYREC OPTIONS=JCL,IF=(1,0,C'VOL=SER='),CHANGE=(1,0,C'UNIT=TAPE3',C'UNIT=TAPE77')
```

The following example will cause STARBAT to reread the data set if the end of the data set was encountered.

```
DD01 COPYREC OPTIONS=MULTI,PRINTHEX=500
```

OR Parameter

Provides for those situations where a number of conditions need to be grouped together with the OR operator. There are two OR syntax forms to test for data contents or valid numerics:

```
OR=(start,length/operator,string,...)     /* data content test   */
OR=(start,length,[duplicate]type,...)     /* valid numeric test  */
```

start	defines the position of the record where the search is to begin. The first byte of the record is position 1.
length	specifies how many characters to search. Specify 0 if you want to search the remainder of the record.
operator	an alternative to length is to specify a relational operator (such as EQ).
string	is a binary, character, hexadecimal or packed string representing data to be searched for at this location.
[duplicate]type	is an optional duplication number followed by **EQN** and **EQP** to check for valid numeric character or packed decimal or **NEN** and **NEP** to check for invalid numeric character or packed decimal.

Note: you may enter multiple parameter sets separated by commas in an OR parameter to obtain multiple logical OR tests.

In this example, if position 40 contains 'COPYRIGHT' or '247B' anywhere from column 2 through the end of the record, copy the record to the file identified in the JCL by FILEA.

```
DD01 MULTICOPY IF=(40,EQ,C'COPYRIGHT'),OR=(2,0,C'247B'),WRITE=FILEA
```

This example copies all records that contain a packed '117400477' or '516988271' in position 11.

```
DD01 COPYREC IF=(11,EQ,{'117400477'),OR=(11,EQ,P'516988271'),MAXRECOUT=0
```

This example copies all records, but if column 1 contains a hex '22722D' OR a hex '28888C', replace column 1 with a hex '99999C'.

```
DD01 COPYALL IF=(1,EQ,X'22722D',1,EQ'X'28888C'),OVERLAY=(1,X'99999C')
```

The example above is equivalent to

```
DD01 COPYALL IF=(1,EQ,X'22722D'),OR=(1,EQ,X'28888C'),OVERLAY=(1,X'99999C')
```

The following example will copy all type 1 records or records with invalid packed numbers to the output data set:

```
DD01 COPYREC IF=(12,EQ,C'1'),OR=(44,0,NEP),PRINTHEX=4
```

This example will copy input records with a 1 in position 10 and a 2 in position 20 or an A in position 3 and a B in position 6. Since it is a COPYREC, only position 30 would be warped; COPYSOME or COPYALL would warp both positions.

```
DD01 COPYREC IF=(10,EQ,C'1'),AND=(20,EQ,C'2'),WARP=(30,C,CCYYMMDD,ADD=5Y),PRINT=3,
            IF=(3,EQ,C'A'),AND=(6,EQ,C'B'),WARP=(40,C,CCYYMMDD,ADD=10D),MAXRECOUT=30
```

OVERLAY or OL Parameter

The OVERLAY parameter replaces data at a particular location with new data in a variety of ways. Data in one location can be overlaid with literal data unconditionally, according to some conditional data content in a location, or according to some conditional data content in another separate location.

There are three OVERLAY syntax forms:

```
OVERLAY=(position,newstring)                                    Replace by location
OVERLAY=(position,length/operator,string-1,string-2)            Replace by condition
OVERLAY=(position,length/operator,string-1,toposition,string-2) Replace at alternate loc
```

position defines the starting position in the output record to which the data is to be moved. Absolute and relative positions can be used. 'n' means at position 'n', '+n' means 'n' bytes further down the record, '-n' means 'n' bytes previous in the record and '+0' means in the next available position.

newstring is a binary, character, hexadecimal or packed string representing data to be placed in the specified location.

length specifies how many characters to search. Specify 0 if you want to search the remainder of the record.

operator an alternative to length is to specify a relational operator (such as EQ).

string-1 is a binary, character, hexadecimal or packed string representing data to be searched for at this location.

toposition defines the starting position in the output record to which the data is to be moved. Absolute and relative positions can be used. 'n' means at position 'n', '+n' means 'n' bytes further down the record, '-n' means 'n' bytes previous in the record and '+0' means in the next available position.

string-2 is a binary, character, hexadecimal or packed string representing data to be replaced at this location.

Caution: OVERLAY replaces existing data with the new data at the specified location without shifting data.

Note that bit operations can be indicated by specifying a modifier as the first character of **string-2**; **O** indicates an OR, **M** indicates NOT-AND or binary minus, **A** indicates AND and **E** indicates an exclusive OR.

The following example will copy all records and place REDRED in the first 6 positions of each record.

```
DD01 COPYREC OVERLAY=(1,2C'RED')
```

The following example will replace the first A found in column 10 through 19 with a B.

```
DD01 COPYREC OVERLAY=(10,10,C'A',C'B')
```

The following example will find RED beginning in position 6 and overlay position 10 with an asterisk.

```
DD01 COPYREC OVERLAY=(6,EQ,C'RED',10,C'*')
```

The following example will clear the zone digits where ABCD is found with an AND bit operation.

```
DD01 COPYREC OVERLAY=(23,20,C'ABCD',AX'0F0F0F0F')
```

OVERALL or OA Parameter

The OVERALL parameter replaces data at a particular location in the same way as OVERLAY does; however, OVERALL replaces all occurrences of the string and it only supports a length field. For more information, see OVERLAY.

There are two OVERALL syntax forms:

```
  OVERLAY=(position,length,string-1,string-2)              Note: Replace by condition
  OVERLAY=(position,length,string-1,toposition,string-2)   Note: Replace at alternate loc
```

position defines the starting position in the output record to which the data is to be moved. Absolute and relative positions can be used. 'n' means at position 'n', '+n' means 'n' bytes further down the record, '-n' means 'n' bytes previous in the record and '+0' means in the next available position.

length specifies how many characters to search. Specify 0 if you want to search the remainder of the record.

string-1 is a binary, character, hexadecimal or packed string representing data to be searched for at this location.

toposition defines the starting position in the output record to which the data is to be moved. Absolute and relative positions can be used. 'n' means at position 'n', '+n' means 'n' bytes further down the record, '-n' means 'n' bytes previous in the record and '+0' means in the next available position.

string-2 is a binary, character, hexadecimal or packed string representing data to be replaced at this location.

The following example will replace all RED's found in columns 10 through 19 with BLU's.

```
DD01 COPYREC OVERALL=(10,10,C'RED',C'BLU')
```

PADCHAR or PAD Parameter

The PADCHAR parameter is used to initialize an output area with the specified character. The default is binary zeros (X'00').

```
PADCHAR=C'c'/X'nn'
```

C'c' c is any single character value.

X'nn' nn is any valid hexadecimal value.

IF a record has 10 blanks beginning in position 11, move the indicated data and pad the end of the records with periods.

```
DD01 COPYREC IF=(11,EQ,10C' '),MOVE=(1,5,28),PADCHAR=C'.'
```

PRINT or P Parameter

The PRINT parameter prints a specified number of records in a simple character format. Note: the PRINT parameter is not intended for use with the print functions; it should be used with any of the other functions that process the data set.

```
   PRINT=n
```

n Any number from 0 to 999999999. Specify 0 to print all records.

The following example will print the first 10 records that contain the string RED anywhere.

```
DD01 COPYREC IF=(1,0,C'RED'),PRINT=10
```

PRINTCHR or PC Parameter

The PRINTCHR parameter prints a specified number of records in an alphanumeric format. Note: PRINTCHR is not intended for use with the print functions; it should be used with any of the other functions that process the data set.

```
   PRINTCHR=n
```

n Any number from 0 to 999999999 can be used. Specify 0 to print all records.

The following example will print the first 10 records that contain the string RED anywhere.

```
DD01 COPYREC IF=(1,0,C'RED'),PRINTCHR=10
```

PRINTHEX or PH Parameter

The PRINTHEX parameter prints a specified number of records in vertical hexadecimal format. Note: PRINTHEX is not intended for use with the print functions; it should be used with any of the other functions that process the data set.

```
   PRINTHEX=n
```

n Any number from 0 to 999999999. Specify 0 to print all records.

The following example will print in vertical hex the first 10 records that contain the string RED anywhere.

```
DD01 COPYREC IF=(1,0,C'RED'),PRINTHEX=10
```

PRINTLPI or PL Parameter

The PRINTLPI parameter controls the number of lines written to a page for the SYSLIST output data set.

```
   PRINTLPI=6/8
```

PRINTLPI=6default, specifies 6 lines to an inch and allows up to 58 lines in a SYSLIST output.
PRINTLPI=8specifies 8 lines to an inch and allows up to 78 lines in a SYSLIST output.

The following example will print the first 10 records containing the string RED with as many as 78 lines on a page.

```
DD01 PRINTHEX IF=(1,0,C'RED'),PRINTCHR=10,PRINTLPI=8
```

RBA Parameter

The RBA parameter is used for VSAM data sets. For ESDS and KSDS, this defines the Relative Byte Address. For RRDS, this defines the Relative Record Number.

```
   RBA=nn/X'nn'
```

nn	nn is any decimal value.
X'nn'	nn is any valid hexadecimal value.

The example below lists records beginning at relative byte address 800. The relative byte address of 800 can also be specified in hex as X'320'.

```
DD01 PRINTCHR RBA=800
```

RDW Parameter

The RDW (Record Descriptor Word) parameter specifies how to address the RECFM=V RDW word.

```
   RDW=0/1/2/3
```

RDW=0 default, the **RDW** word is to be included for record processing and output displays.
RDW=1 the **RDW** word is to be included for record processing but not for output displays.
RDW=2 the **RDW** word is not to be included for record processing but is to be shown for output displays.
RDW=3 the **RDW** word is not to be included for record processing and not to be shown for output displays.

The following example lists the first 10 records that contain RED in the first data position and the RDW word is to be included for record processing and output:

```
DD01 PRINTCHR RDW=0,MAXRECOUT=10,IF=(5,EQ,C'RED')
```

The following example is similar, looking for records with RED in the first data position. Note that the RDW word is not to be displayed in the output:

```
DD01 PRINTCHR RDW=3,MAXRECOUT=10,IF=(1,EQ,C'RED')
```

SELECT or S Parameter

The SELECT parameter may be used to select every nth occurrence of a record for processing.

```
   SELECT=n
```

n is the number of records in an interval. Any number from 1 to 999999999 may be used.

The following example copies all records and tests every third record for a 'PBCE' in column 3. If a match is found, 'PBCE' is replaced by four zeroes.

```
DD01 COPYALL SELECT=3,OVERLAY=(3,EQ,C'PBCE',C'0000')
```

The following example only processes records containing 'Z145' in column 3. Every third record satisfying the conditional statement will be copied and have 'Z145' replaced with 'ZZZZ'.

```
DD01 COPYREC IF=(3,EQ,C'Z145'),SELECT=3,OVERLAY=(3,C'ZZZZ')
```

In this example, every third record is selected, and if the selected record contains a 'PBCE' in column 3, copy the record and replace 'PBCE' with 'ZZZZ'.

```
DD01 COPYREC SELECT=3,IF=(3,EQ,C'PBCE'),OVERLAY=(3,C'ZZZZ')
```

STARTKEY or SK Parameter

The STARTKEY parameter can be specified to initiate VSAM KSDS processing at a specific or generic key.

```
   STARTKEY=C'cc'/X'hh'
```

STARTKEY=C'cc' is any character string that specifies the beginning KSDS key value.
STARTKEY=X'hh' is any hexadecimal string that specifies the beginning KSDS key value.

The following example will copy VSAM records after positioning to generic key RED:

```
DD01 COPYREC STARTKEY=C'RED'
```

STOPIF or ST Parameter

Use the STOPIF parameter if you want to terminate processing before the end of data. STOPIF will end execution of the current process if the condition associated with the parameter is satisfied.

```
   STOPIF=(start,length/operator,string,...)
```

start defines the position of the record where the search is to begin. The first byte of the record is position 1.
length specifies how many characters to search. Specify 0 if you want to search the remainder of the record.
operator an alternative to length is to specify a relational operator (such as EQ).
string is a binary, character, hexadecimal or packed string representing data to be searched for at this location.

Note: You may code multiple conditions within one STOPIF parameter; these are evaluated using logical OR tests. Multiple contiguous STOPIF parameters result in logical AND tests for STOPIF conditions.

The following example will copy all records until 'RECORDS READ' is found in position 8. The record containing 'RECORDS READ' is not copied.

```
DD01 COPYREC STOPIF=(8,EQ,C'RECORDS READ')
```

In this example, the first DD01 control statement will copy records until VOL001 is found in column 1. The second DD01 control statement will begin with the VOL001 record and stop copying records with VOL300.

```
DD01 COPYREC STOPIF=(1,EQ,C'VOL001')
DD01 COPYREC STOPIF=(1,EQ,C'VOL300')
```

The following example will skip to the record containing a hex '033198' in column 1 or 'IMAGE' in column 12. This record and all records after it are copied.

```
DD01 SKIP STOPIF=(1,EQ,X'033198',12,EQ,C'IMAGE')
DD01 COPYREC
```

SUM Parameter

The SUM parameter is used to add up any fields containing character, packed, or binary numbers. If there is a SYSTOTAL data set, then the totals will go to that data set. Otherwise, they will go to the SYSPRINT data set

```
   SUM=(position[,'description])                  /* for packed decimal data      */
   SUM=(position,length,datatype[,'description'])  /* for character or binary data */
```

position The beginning position of the data. This can be an actual or a relative start position.
length The length of the data to be totaled. For character data, specify a length of 1 to 15; for binary data, specify a length of 1 to 4; for packed data, its length is calculated by STARBAT.
datatype The type of data. For character data, specify **C**; for binary data, specify **B**; for packed data, you do not need to specify the type.
description This is optional. A maximum of 25 characters enclosed in single quotes can be entered. If a description is present, it will be displayed as report headers; otherwise, the control statement will be used as a report header.

The following example selects records that contain a packed decimal '9802' in position 5, then adds the packed field at position 12. These selected records are printed in vertical format in hex.

```
DD01 PRINTHEX IF=(5,EQ,P'9802'),SUM=(12,'HIST TOTAL')
```

The example below begins processing a VSAM KSDS data set with the record at location x'00433C' and then adds the data at position 73. Processing stops if the key is greater than x'00433C'.

```
DD01 TOTAL KEY=X'00433C',
          STOPIF=(1,GT,X'00433C'),
          SUM=(073,8,C,'TRAVEL EXPENSE REPORT')
```

WARP Parameter

The WARP parameter may be used to modify numeric data values and dates in many formats. Note that the WARP parameter has three required positional parameters and multiple optional keyword parameters. The WARPDEF parameter may be used to set defaults for the WARP parameter.

The WARP parameter is described in two sections. This first section describes warping numeric data and converting currencies like the Euro; the following section describes warping dates.

Warping Numeric Data

To warp numeric fields, STARBAT first performs an **ADD**, **SUB**, **DIV** or **MULT** operation. If a **CONV** subparameter is specified, conversion values are obtained from an exit table and both a **DIV** and **MULT** operation can be performed. After the calculation is performed, any **ACTION** is performed. Finally, the modified data is output using the **OUTLOC** location, **OUTPIC** picture and **OUTFORM** format.

```
WARP=(location,type,picture,
      ADD=nn,SUB=nn,DIV=nn,MULT=nn,CONV=typcnv,
      ACTION=aa,OUTLOC=loc,OUTFORM=type,OUTPIC=picture)
```

location is the location where the data element begins. An actual or relative location may be used.

type is the storage format of the data element; this can be **B**, **C** or **P** representing binary, character or packed.

picture describes the data item. **S9**, **9**, **S99**, **99** and other variations with up to 15 contiguous 9 digits are supported for numeric data values. Note that a single **V** character can also be specified to indicate the logical position of the decimal point; otherwise, it is assumed after the last digit.

ADD=nn requests an addition operation where **nn** is a number with an optional decimal point.

SUB=nn requests a subtraction operation where **nn** is a number with an optional decimal point.

DIV=nn requests a division operation where **nn** is a number with an optional decimal point. Note that a division is normally required to convert the currency of an European member state to Euro units.

MULT=nn requests a multiplication operation where **nn** is a number with an optional decimal point. Note that a multiplication is normally required to convert Euro units to the currency of an European member state.

CONV=typcnv requests that conversion values be looked up in a conversion table. The **typcnv** token is normally composed of two consecutive parts: **typ** is a three character code indicating the beginning currency and **cnv** is a three letter code indicating the target currency. The **CONV** subparameter invokes a conversion exit named PDS#CONV; this exit is supplied in source and load form with STARWARP and it has initial conversion factors supplied in table format.

Note that conversions from the currency of a European member state to the currency of a different member state normally requires a divide to convert to Euro units and a multiply to convert to the target currency. Rounding is only performed after both operations have completed.

The **typ** and **cnv** letter codes can be any of the following:

EUR	Euro units
BEF	Belgian Franc (smallest currency unit is 1.00)
LUF	Luxembourg Franc (smallest currency unit is 1.00)
DEM	Deutsche Mark
ESP	Spanish Peseta (smallest currency unit is 1.00)
FRF	French Franc
IEP	Irish Punt
ITL	Italian Lira (smallest currency unit is 1.00)

NLG	Netherlands Guilder
ATS	Austrian Schilling
PTE	Portuguese Escudo (smallest currency unit is 0.10)
FIM	Finish Markka
GRD	Greek Drachma (was not eligible to join initially)
DKK	Danish Krone (did not join initially)
SEK	Swedish Krona (did not join initially)
GBP	United Kingdom Pound (did not join initially)
USD	United States Dollar (not a member)
CAD	Canadian Dollar (not a member)
ASD	Australian Dollar (not a member)
JAY	Japanese Yen (not a member)

ACTION=aa requests the type of data handling desired after the numeric operation.

ODROP	drop a record if a data overflow occurs for a numeric element
OFLAG	issue a message if a data overflow occurs for a numeric element
OZERO	convert a data overflow for a numeric element to a zero value
OMAX	convert a data overflow for a numeric element to a maximum value

OUTLOC=l is the location where the output data element is to be placed. This keyword defaults to the value used for the input location and an actual or relative location may be used.

OUTFORM=f is the storage format of the output data element; this can be **B**, **C** or **P** representing binary, character or packed respectively and it defaults to the format specified for input. Note that **OUTFORM=** and **OUTPIC=** will not cause data positions to be shifted.

OUTPIC=p is the picture of the output data; this value defaults to the input data picture but you may specify any of the picture formats supported for input. Note that **OUTPIC=** and **OUTFORM=** will not cause data positions to be shifted.

The following example will select all records with a 1 in column 10 and warp the binary data by adding 1147.

```
DD01 COPYREC IF=(10,EQ,C'1'),WARP=(27,B,999999999,ADD=1147)
```

The following example will select all records with a 1 in column 10 and add 10% to each item selected by multiplying:

```
DD01 COPYREC IF=(10,EQ,C'1'),WARP=(27,P,99999999999V99,MULT=1.1)
```

The following example will convert Italian lira to Euros and save the result in another field.

```
DD01 COPYREC WARP=(27,C,999999999999,DIV=1906.48,OUTPIC=999999999V99,OUTLOC=44)
```

The following example is equivalent but it uses a conversion table:

```
DD01 COPYREC WARP=(27,C,999999999999,CONV=ITLEUR,OUTPIC=999999999V99,OUTLOC=44)
```

The following example will convert Euros to Italian lira and save the result in another field.

```
DD01 COPYREC WARP=(122,P,9999999999V99,MULT=1906.48,OUTPIC=99999999999,OUTLOC=145)
```

The following example will convert Italian lira to German marks using a conversion table:

```
DD01 COPYREC WARP=(145,P,999999999999,CONV=ITLDEM,OUTPIC=999999999V99,OUTLOC=208)
```

Warping Dates

To warp dates, STARBAT first performs a **DATE**, **ADD** or **SUB** parameter followed by **ACTION** and **VALID.** Finally, the date is output using the **OUTLOC** location, **OUTPIC** picture and **OUTFORM** format.

```
WARP=(location,type,picture,
      DATE=any,ADD=nni,SUB=nni,
      IGNORE=blhz9pi,FISCAL=mm,PIVOTYR=nn,BASEYR=mmmm,ERRWRITE=ddname,
      ACTION=aa,VALID=vv,HOLIDAY=name,
      OUTLOC=loc,OUTFORM=type,OUTPIC=picture)
```

location is the location where the data element begins. An actual or relative location may be used.

type is the storage format of the data element; this can be **B**, **C** or **P** representing binary, character or packed.

picture describes the date. A date picture may be composed of **CCYY**, **CYY**, **HYY**, **YY**, **MM**, **DDD** or **DD** components in any order in any of the following basic combinations:

1. **CCYY**, **CYY**, **HYY** or **YY**
2. **CCYY**, **CYY**, **HYY** or **YY** with **DDD**
3. **CCYY**, **CYY**, **HYY** or **YY** with **MM**
4. **CCYY**, **CYY**, **HYY** or **YY** with **MM** and **DD**

Notes:

1. **HYY**... pictures use a century digit of 0 for 19xx dates and 1 for 20xx dates; 1998/01/01 is represented as 0980101 with a HYYMMDD picture.
2. **CYY...** pictures use a century digit of 1 for 19xx dates and 2 for 20xx dates; 1998/01/01 is represented as 1980101 with a **CYYMMDD** picture.
3. For dates that contain all numeric characters, **CC99**, **C99**, **H99** and **99** may be used instead of the year values to specify that nine's compliment dates are to be used; 1998/01/01 is represented as 80019898 with a **CC99MMDD** picture.
4. For character dates, **MON** may be used interchangeably with **MM**. Pictures containing **MON** use a three-character month abbreviation such as JAN or FEB; 1998/01/01 is represented as 01JAN98 with a **DDMONYY** picture.
5. For character dates, up to ten separator characters may be added between the individual date components described above. Slash, period, blank (indicated by a **B**) and minus characters may be used as separators; in addition, a question mark character may be used to indicate that any character is acceptable at a location. For example, 1998/01/01 is represented as 01/01/98 with a **MM/DD/YY** picture, 01.01.1998 with a **MM.DD.CCYY** picture, or 01-JAN-1998 with a **DD-MON-CCYY** picture.

In addition to the above pictures, the following are also supported: **LLLLLL**, **COBLLL, MM, MMDD, YDDD, YMMDD** and any name beginning with **PDS**. Note the following:

1. **LLLLLL** represents an integer Lilian date where day 1 was Friday, October 15, 1582 and November 1, 1996 is 151229.
2. **COBLLL** represents an integer COBOL Lilian date where day 1 was January 1, 1600 and November 1, 1996 is 144942 (or the Lilian date - 6287 days).
3. **MM** and **MMDD** pictures assume year values from the current system date.
4. **YDDD** and **YMMDD** pictures use the second year digit and are always assumed to be dates in the future. This means that 8001 with a YDDD picture would represent 1998/01/01 before that date and it would represent 2008/01/01 after 1998/01/01.
5. A picture beginning with **PDS** names an input/output exit for date conversion. This exit routine will be called for the initial conversion of a date to internal format and to format the final date unless OUTPIC is specified. Sample exits with additional details have been written to support date formats like "January 1, 1998" and A0MMDD pictures. Contact SERENA for more details on available date conversion exits.

DATE=any requests that the identified field be set to the date specified; this date must be entered in the same format as the date picture chosen.

ADD=nni requests an addition operation where **nn** is a number and **i** is

- **Y** for years (0Y to 999Y is permitted)
- **M** for months (0M to 99M is permitted)
- **W** for weeks (0W to 999W is permitted)
- **D** for days (0D to 9999D is permitted).
- **B** for business days (0B to 9999B is permitted -- use this with HOLIDAY=)

SUB=nni requests a subtraction operation where **nn** is a number and **i** is

- **Y** for years (0Y to 999Y is permitted)
- **M** for months (0M to 99M is permitted)
- **W** for weeks (0W to 999W is permitted)
- **D** for days (0D to 9999D is permitted).
- **B** for business days (0B to 9999B is permitted -- use this with HOLIDAY=)

IGNORE=code requests that certain error or special value dates are to be ignored or bypassed. The code letters for IGNORE may be entered in any order from the following set:

- **B** ignores blank dates
- **L** ignores low values (all X'00' characters)
- **H** ignores high values (all X'FF' characters)
- **Z** ignores all zero dates
- **9** ignores all nine dates (such as 9/9/99 and Julian 1999.099 or 1999/04/09)
- **P** ignores permanent retention dates (such as 99365 in a YYDDD picture)
- **I** ignores invalid input dates (such as February 30, 1998)

FISCAL=mm defaults to 1 and may be specified as 1 through 12 to indicate the start of your fiscal year for ACTION processing keywords like FQEND or FYEND.

PIVOTYR=nn defaults to 70 and may be specified as 0 through 99 to indicate the pivot year for windowing two digit year values. For example, if PIVOTYR=60, a year value of 59 is interpreted as 2059 and a year value of 61 is interpreted as 1961. A year value equal to the PIVOTYR value will also be translated to the next century.

Note that the data value can also be entered as a negative number to define a sliding year window. For example, **PIVOTYR=-20** indicates that pivot year begins 20 years before the current date.

BASEYR=iiii defaults to 1900 and provides century digits for windowing two digit dates. **BASEYR** may be entered as 1500 through 9800 and it should always be a multiple of 100.

BASEYR and **PIVOTYR** are used together to window two digit dates as described below:

```
CCYYDATE=datevalue
IF CCYYDATE<100 THEN DO
    IF CCYYDATE<=PIVOTYR THEN CCYYDATE=CCYDATE+100
    CCYYDATE=CCYYDATE+BASEYR
    END
```

ERRWRITE=d specifies a DDNAME to output records that contain invalid date values for a WARP parameter. If the ERRWRITE parameter is present, the current record will be written to this data set if the date is considered invalid by STARBAT or a STARBAT date exit. If you warp a record has multiple times, you could get multiple copies of the record written to the various ERRWRITE data sets.

HOLIDAY=h names a holiday table that is to be called for business day calculations for **ADD=nnB**, **ACTION** and **VALID** above (**MFBD**, **QFBD**, **YFBD**, **MLBD**, **QLBD**, **YLBD**, **NEXTBDAY** and **PREVBDAY**) to determine business holidays. The holiday routine is a passed a date and it must return a code to indicate if that date is a holiday, not a holiday or some error occurred. A sample holiday routine is provided that contains all U.S. holidays between 1990 and 2020; this exit contains additional instructions so that the calendars can be modified for other countries or a specific business use. Note that if a holiday routine is not requested, all weekdays are assumed to be business days.

ACTION=aa requests the type of date or data handling desired after **DATE**, **ADD** or **SUB**. Note that the **NEXT..** and **PREV..** actions below check the derived date for validity (for example: February 31) and move any invalid dates to the end of the current month before taking any action.

MEND	move a date to the end of the month
QEND	move a date to the end of the quarter
YEND	move a date to the end of the year
MFBD	move a date to the first business day of this month
QFBD	move a date to the first business day of this quarter
YFBD	move a date to the first business day of this year
MLBD	move a date to the last business day of this month
QLBD	move a date to the last business day of this quarter
YLBD	move a date to the last business day of this year
FQEND	move a date to the end of the fiscal quarter; use with FISCAL=
FYEND	move a date to the end of the fiscal year; use with FISCAL=
FQFBD	move a date to the first business day of this fiscal quarter; use with FISCAL=
FYFBD	move a date to the first business day of this fiscal year; use with FISCAL=
FQLBD	move a date to the last business day of this fiscal quarter; use with FISCAL=
FYLBD	move a date to the last business day of this fiscal year; use with FISCAL=
NEXTBDAY	move a date to the next business day if it is not already a business day
PREVBDAY	move a date to the previous business day if it is not already a business day
BALIGN	move a date to the last business day of the month if it started as such for a month
NEXTFIRST	move a date after the first of the month to the first of the next month
PREVFIRST	move a date after the first of the month back to the first of this month
NEXTWKDAY	move a weekend date to the next Monday if not already at Monday
PREVWKDAY	move a weekend date to the previous Friday if not already at Friday
NEXTday	move a date to the next **day** (where **day** is **MON**day, **TUE**sday, **WED**nesday, **THU**rsday, **FRI**day, **SAT**urday or **SUN**day) if not already at **day**
PREVday	move a date to the previous **day** (where **day** is **MON**day, **TUE**sday, **WED**nesday, **THU**rsday, **FRI**day, **SAT**urday or **SUN**day) if not already at **day**
NEXTSDAY	move a date to the next same day of week if it is not already the same day of week
PREVSDAY	move a date to the previous same day of week if it is not already the same day of week
DROP	drop the record if the derived date is invalid like February 30
FLAG	issue a message if the derived date is invalid like February 30
ROLL	move the date into the next month if the derived date is invalid
ADJUST	move the date to the end of the month if originally at month end or the date is invalid
ALIGN	move the date to the end of the month if the date was at month end originally
dow#	move the date to the nth day of week in the month where dow is MON, TUE, WED, THU, FRI, SAT or SUN and # is 1, 2, 3, 4 or LAST.
dow#B	move to the nth day of week and next business day
dow#P	move to the nth day of week and previous business day
DAY#	move the date to a day in the month (1 or 01 through 31)
DAY#B	move the date in the month and next business day
DAY#P	move the date in the month and previous business day

VALID=vv requests the type of date validation desired after making after a **DATE**, **ADD** or **SUB** change, and any **ACTION** adjustment. Note that **VALID** is normally only used to check dates that are already present in a data set and it is normally used without using **DATE**, **ADD**, **SUB** or **ACTION** at the same time.

WKEND	verify the date is valid and falls on a Saturday or Sunday
WKDAY	verify the date is valid and falls between Monday through Friday
MEND	verify the date is valid and falls on the last day of the month
QEND	verify the date is valid and falls on the last day of the quarter
YEND	verify the date is valid and falls on the last day of the year
MFBD	verify the date is valid and falls on the first business day of the month
QFBD	verify the date is valid and falls on the first business day of the quarter
YFBD	verify the date is valid and falls on the first business day of the year
MLBD	verify the date is valid and falls on the last business day of the month
QLBD	verify the date is valid and falls on the last business day of the quarter
YLBD	verify the date is valid and falls on the last business day of the year

OUTLOC=l is the location where the output date element is to be placed. This keyword defaults to the value used for the input location and an actual or relative location may be used.

OUTFORM=f is the storage format of the output date element; this can be **B**, **C** or **P** representing binary, character or packed respectively and it defaults to the format specified for input. Note that **OUTFORM=** and **OUTPIC=** will not cause data positions to be shifted.

OUTPIC=p is the picture of the output date; this value defaults to the input date picture but you may specify any of the picture formats supported for input. Note that character formats like **YY/MM/DD** are supported if **OUTFORM=C** is specified or is the default. Also, note that **OUTPIC=** and **OUTFORM=** will not cause data positions to be shifted.

Warp Logic

1. The date **picture** implicitly defines the number of storage positions. For binary **type** values, four characters are always used; for packed dates, (n+1)/2 characters are used, where n is the number of digits; and for character dates, the picture defines the number of characters used.
2. From the **location**, the date is converted to an internal format (in Gregorian, Julian and Lilian formats).
3. A **DATE=**, **ADD=** or **SUB=** operation is performed to warp the date.
4. **ACTION=** is performed using the **HOLIDAY=** exit if available for business day calculations like **MFBD**.
5. **VALID=** is performed to validate dates using any **HOLIDAY=** exit as above.
6. The date is output at the location specified by **OUTLOC=** in format **OUTFORM=** using picture **OUTPIC=**. Note that **OUTFORM=** and **OUTPIC=** specifications will not cause data positions to be shifted.

The following example will select all records with a 1 in column 10 and replace the data with a fixed date.

```
DD01 COPYREC IF=(10,EQ,C'1'),WARP=(27,C,CCYYMMDD,DATE=20010112)
```

The following example will select all records with a 1 in column 10 and modify associated packed dates by adding 2 months and verifying that the date is on the last day of the month.

```
DD01 COPYREC IF=(10,EQ,C'1'),WARP=(27,P,CCYYMMDD,ADD=2M,VERIFY=MEND)
```

The following example will select all records with a 1 in column 10 and 2 in column 20 and modify associated dates by adding 5 years and moving the resulting date to a weekday if necessary.

```
DD01 COPYREC IF=(10,EQ,C'1'),AND=(20,EQ,C'2'),
       WARP=(27,C,CCYYMMDD,ADD=5Y,ACTION=NEXTWKDAY)
```

The following example will select all records with a 1 in column 10 (and modify associated dates by adding five years) or 2 in column 20 (and modify associated dates by adding 10 days). Note that if the first condition is successful, no additional conditions are tested. Use **COPYSOME** to copy selected records and perform both tests.

```
DD01 COPYREC    IF=(10,EQ,C'1'),WARP=(27,C,CCYYMMDD,ADD=5Y),
               OR=(20,EQ,C'2'),WARP=(37,C,YYDDD,ADD=10D)
```

The following example will expand a date field and convert the date to a different format.

```
DD01 COPYREC EXPAND=(3,C,6,10),WARP=(3,C,YYMMDD,OUTPIC=CCYY/MM/DD)
```

The following example will modify dates by adding 5 years and move the resultant date to the next business day using the WORKSCHD holiday exit.

```
DD01 COPYREC WARP=(27,C,CCYYMMDD,ADD=5Y,ACTION=NEXTBDAY,HOLIDAY=WORKSCHD)
```

The following example will add 10 business days to each date.

```
DD01 COPYREC WARP=(27,C,CCYYMMDD,ADD=10B,HOLIDAY=WORKSCHD)
```

The following example will move a date to the next fiscal quarter and add 10 business days by warping the date twice.

```
DD01 COPYREC WARP=(27,C,CCYYMMDD,ADD=3M,FISCAL=2,ACTION=FQFBD,HOLIDAY=WORKSCH,OUTLOC=55),
            WARP=(55,C,CCYYMMDD,ADD=10B,HOLIDAY=WORKSCHD)
```

The following example will modify the same date field in different ways to produce two output date fields.

```
DD01 COPYREC WARP=(22,C,CCYYMMDD,ADD=5M,OUTLOC=34,OUTPIC=CCYY/MM/DD),
            WARP=(22,C,CCYYMMDD,ADD=9M,OUTLOC=44,OUTPIC=CCYY/MM/DD)
```

The following example will warp a date and write records with invalid dates to `//DDOUT`:

```
DD01 COPYREC WARP=(27,C,CCYYMMDD,ADD=5Y,OUTLOC=39,OUTPIC=CCYY/MM/DD,ERRWRITE=DDOUT)
```

WARPDEF or WD Parameter

The WARPDEF parameter may be used to define default values for the WARP parameter. Subparameters defined with WARPDEF stay in effect until they are redefined or nullified; a WARP parameter may also override the WARPDEF default.

```
WARPDEF=(ADD=nni,SUB=nni,
       IGNORE=blhz9pi,FISCAL=mm,PIVOTYR=nn,BASEYR=mmmm,
       ACTION=aa,VALID=vv,HOLIDAY=name)
```

ADD=nni requests an addition operation where **nn** is a number and **i** is

- **Y** for years (0Y to 999Y is permitted)
- **M** for months (0M to 99M is permitted)
- **W** for weeks (0W to 999W is permitted)
- **D** for days (0D to 9999D is permitted).
- **B** for business days (0B to 9999B is permitted -- use this with HOLIDAY=)

SUB=nni requests a subtraction operation where **nn** is a number and **i** is

- **Y** for years (0Y to 999Y is permitted)
- **M** for months (0M to 99M is permitted)
- **W** for weeks (0W to 999W is permitted)
- **D** for days (0D to 9999D is permitted).
- **B** for business days (0B to 9999B is permitted -- use this with HOLIDAY=)

IGNORE=code requests that certain error or special value dates are to be ignored or bypassed. The code letters for IGNORE may be entered in any order from the following set:

- **B** ignores blank dates
- **L** ignores low values (all X'00' characters)
- **H** ignores high values (all X'FF' characters)
- **Z** ignores all zero dates
- **9** ignores all nine dates (such as 9/9/99)
- **P** ignores permanent retention dates (such as 99365 in a YYDDD picture)
- **I** ignores invalid input dates (such as February 30, 1998)

FISCAL=mm defaults to 1 and may be specified as 1 through 12 to indicate the start of your fiscal year for ACTION processing keywords like FQEND or FYEND.

PIVOTYR=nn defaults to 70 and may be specified as 0 through 99 to indicate the pivot year for windowing two digit year values. For example, if PIVOTYR=60, a year value of 59 is interpreted as 2059 and a year value of 61 is interpreted as 1961. A year value equal to the PIVOTYR value will also be translated to the next century. Note that the data value can also be entered as a negative number to define a sliding year window. For example, **PIVOTYR=-20** indicates that pivot year begins 20 years before the current date.

BASEYR=iiii defaults to 1900 and it provides century digits for windowing two digit dates. **BASEYR** may be entered as 1500 through 9800 and it should always be a multiple of 100.

BASEYR and **PIVOTYR** are used together to window two digit dates as described below:

```
CCYYDATE=datevalue
IF CCYYDATE<100 THEN DO
    IF CCYYDATE<=PIVOTYR THEN CCYYDATE=CCYDATE+100
    CCYYDATE=CCYYDATE+BASEYR
    END
```

ACTION=aa requests the type of date or data handling desired after **DATE**, **ADD** or **SUB**. Note that the **NEXT..** and **PREV..** actions below check the derived date for validity (for example: February 31) and move any invalid dates to the end of the current month before taking any action.

MEND	move a date to the end of the month
QEND	move a date to the end of the quarter
YEND	move a date to the end of the year
MFBD	move a date to the first business day of this month
QFBD	move a date to the first business day of this quarter
YFBD	move a date to the first business day of this year
MLBD	move a date to the last business day of this month
QLBD	move a date to the last business day of this quarter
YLBD	move a date to the last business day of this year
FQEND	move a date to the end of the fiscal quarter; use with FISCAL=
FYEND	move a date to the end of the fiscal year; use with FISCAL=
FQFBD	move a date to the first business day of this fiscal quarter; use with FISCAL=
FYFBD	move a date to the first business day of this fiscal year; use with FISCAL=
FQLBD	move a date to the last business day of this fiscal quarter; use with FISCAL=
FYLBD	move a date to the last business day of this fiscal year; use with FISCAL=
NEXTBDAY	move a date to the next business day if it is not already a business day
PREVBDAY	move a date to the previous business day if it is not already a business day
BALIGN	move a date to the last business day of the month if it started as such for a month
NEXTFIRST	move a date after the first of the month to the first of the next month
PREVFIRST	move a date after the first of the month back to the first of this month
NEXTWKDAY	move a weekend date to the next Monday if not already at Monday
PREVWKDAY	move a weekend date to the previous Friday if not already at Friday
NEXTday	move a date to the next **day** (where **day** is **MON**day, **TUE**sday, **WED**nesday, **THU**rsday, **FRI**day, **SAT**urday or **SUN**day) if not already at **day**
PREVday	move a date to the previous **day** (where **day** is **MON**day, **TUE**sday, **WED**nesday, **THU**rsday, **FRI**day, **SAT**urday or **SUN**day) if not already at **day**
NEXTSDAY	move a date to the next same day of week if it is not already the same day of week
PREVSDAY	move a date to the previous same day of week if it is not already the same day of week
DROP	drop the record if the derived date is invalid like February 30
FLAG	issue a message if the derived date is invalid like February 30
ROLL	move the date into the next month if the derived date is invalid
ADJUST	move the date to the end of the month if originally at month end or the date is invalid
ALIGN	move the date to the end of the month if the date was at month end originally
dow#	move the date to the nth day of week in the month where dow is MON, TUE, WED, THU, FRI, SAT or SUN and # is 1, 2, 3, 4 or LAST.
dow#B	move to the nth day of week and next business day
dow#P	move to the nth day of week and previous business day
DAY#	move the date to a day in the month (1 or 01 through 31)
DAY#B	move the date in the month and next business day
DAY#P	move the date in the month and previous business day

VALID=vv requests the type of date validation desired after making after a **DATE**, **ADD** or **SUB** change, and any **ACTION** adjustment. Note that **VALID** is normally only used to check dates that are already present in a data set and it is normally used without using **DATE**, **ADD**, **SUB** or **ACTION** at the same time.

WKEND	verify the date is valid and falls on a Saturday or Sunday
WKDAY	verify the date is valid and falls between Monday through Friday
MEND	verify the date is valid and falls on the last day of the month
QEND	verify the date is valid and falls on the last day of the quarter
YEND	verify the date is valid and falls on the last day of the year
MFBD	verify the date is valid and falls on the first business day of the month
QFBD	verify the date is valid and falls on the first business day of the quarter
YFBD	verify the date is valid and falls on the first business day of the year
MLBD	verify the date is valid and falls on the last business day of the month
QLBD	verify the date is valid and falls on the last business day of the quarter
YLBD	verify the date is valid and falls on the last business day of the year

HOLIDAY=h names a holiday table that is to be called for business day calculations for **ADD=nnB**, **ACTION** and **VALID** above (**MFBD**, **QFBD**, **YFBD**, **MLBD**, **QLBD**, **YLBD**, **NEXTBDAY** and **PREVBDAY**) to determine business holidays. The holiday routine is a passed a date and it must return a code to indicate if that date is a holiday, not a holiday or some error occurred. A sample holiday routine is provided that contains all U.S. holidays between 1990 and 2020; this exit contains additional instructions so that the calendars can be modified for other countries or a specific business use. Note that if a holiday routine is not requested, all weekdays are assumed to be business days.

The following example will set a default of 5 years and warp two date fields:

```
DD01 COPYREC WARPDEF=ADD=5Y,WARP=(27,C,CCYYMMDD),WARP=(107,P,CCYYMMDD)
```

The following example is similar but it warps the second date by 10 years:

```
DD01 COPYREC WARPDEF=ADD=5Y,WARP=(27,C,CCYYMMDD),WARP=(107,P,CCYYMMDD,ADD=10Y)
```

The following example is more complex in that it sets several defaults and nullifies them later:

```
DD01 COPYREC WARPDEF=(ACTION=NEXTBDAY,HOLIDAY=PDS#HOLI,ADD=5Y),
            WARP=(107,P,CCYYMMDD),WARP=(122,P,CCYY),
            WARPDEF=(ACTION=,HOLIDAY=,ADD=1Y),
            WARP=(6,C,CCYY/DDD)
```

WRITE or W Parameter

The WRITE parameter is used only by the **MULTICOPY** function to control when output is written to the named DD statements. The WRITE parameter can direct output to any number of output DD statements; if the same DDNAME is used more than once, multiple copies of the output record will be written.

There are several restrictions on the data that can be output with a **MULTICOPY** function:

1. Output to load members is supported; however JCL indicating PDS(MEMBER) notation is not supported. Instead, use only the PDS name in the JCL and indicate the member name with the MEMBER, MEMBERS, NEWMBR or NEWMBRS parameters.
2. Overlay, scatter-loaded and note-listed load members are not supported.

```
   WRITE=ddname1/(ddname1,ddname2,...)
```

ddnamex the DDNAME in the JCL containing the name of the output data set.

The following example shows how to create two identical output data sets in **//OUT1** and **//OUT2**:

```
DD01 MULTICOPY WRITE=(OUT1,OUT2)
```

The following example will copy a file and repeat any record with a C'5' and change it to a C'6':

```
DD01 MULTICOPY WRITE=OUT1,IF=(20,EQ,C'5'),MOVE=(1,0,1),
            MOVE=(20,C'6'),WRITE=OUT1
```

The following example will add a STEPLIB JCL statement after any EXEC statement:

```
DD01 MULTICOPY WRITE=NEWJCL,IF=(1,20,C' EXEC '),MOVE=(1,80C' '),
            MOVE=(1,C'//STEPLIB  DD  DISP=SHR,'),
            MOVE=(+0,C'DSN=MYHILEV.MYMIDLEV.MYLOWLEV'),WRITE=NEWJCL
```

STARBAT Logic

Logical AND Conditions

Consecutive IF parameters represent a logical AND condition. In other words,
`IF=(1,EQ,C'A'),AND=(10,EQ,C'B')` has the same effect as `IF=(1,EQ,C'A'),IF=(10,EQ,C'B').`

This example will copy all records containing TEST RECORD in position 23 and RED anywhere after position 10:

```
DD01 COPYREC IF=(23,EQ,C'TEST RECORD'),IF=(10,0,C'RED')
```

This example is equivalent because AND and IF are identical in meaning:

```
DD01 COPYREC IF=(23,EQ,C'TEST RECORD'),AND=(10,0,C'RED')
```

Logical OR Conditions

Multiple data comparisons can be coded within an IF, AND, or OR parameter. A logical OR condition is assumed between multiple data comparisons; logical OR conditions can be coded in multiple different formats as shown below.

The following example will copy all records containing TEST RECORD in position 23 or RED anywhere after position 10:

```
DD01 COPYREC IF=(23,EQ,C'TEST RECORD',10,0,C'RED')
```

The following example is identical in effect to the previous example:

```
DD01 COPYREC IF=(23,EQ,C'TEST RECORD'),OR=(10,0,C'RED')
```

The following example will copy up to 10 records containing TEST RECORD in position 23 and RED anywhere after position 10 or the first 20 records containing A in position 3 and B in position 6:

```
DD01 COPYREC IF=(23,EQ,C'TEST RECORD'),AND=(10,10,C'RED'),MAXRECOUT=10,
             IF=(3,EQ,C'A'),AND=(6,EQ,C'B'),MAXRECOUT=20
```

Processing Multiple Selection Parameters

The IF, AND, and OR parameters are called **selection** parameters because they are used to select records to be processed. **Limit** parameters like MAXRECIN and MAXRECOUT can follow the selection parameters to apply processing record limits and **action** parameters like CHANGE or WARP may be specified to modify or inspect data fields. In addition, **print** parameters like PRINTHEX may be specified to selectively print records and **control** parameters like ABEND or STARTKEY can be used to select basic execution choices.

Complex selection criteria can be specified using multiple selection parameters. Multiple selection parameters are handled in groups where a group of selection parameters starts with the first selection parameter and ends after any associated limit, action and print parameters that follow the selection parameters.

For example;

```
DD01 COPYREC IF=(10,EQ,C'1'),AND=(20,EQ,C'2'),WARP=(30,C,CCYYMMDD,ADD=5Y),PRINT=3,
           IF=(3,EQ,C'A'),AND=(6,EQ,C'B'),WARP=(40,C,CCYYMMDD,ADD=10D),MAXRECOUT=30
```

In this example, there are two groups of selection parameters. The first group consists of the first IF parameter and the first AND parameter and is followed by the first WARP action parameter and the PRINT parameter. The second group consists of the second IF parameter and the second AND parameter and is followed by the second WARP action parameter and the MAXRECOUT limit parameter.

A logical OR condition is assumed between groups of selection parameters that are followed by limit, action or print parameters. This means that if an input record contains data that matches any one of the groups of selection parameters then that input record is selected for processing. In the example above, records that contain a 1 in column 10 and a 2 in column 20 are copied as well as records that contain an A in column 3 and a B in column 6.

If an input record contains data that matches with a group of selection parameters, then the limit, action and print parameters that follow that group are processed and STARBAT skips to the next processing group. Whether STARBAT actually processes those parameters depends on the name of the current function. If the function has an **ALL** or **SOME** suffix as a part of its name (for example, **COPYSOME** or **COPYALL** versus **COPYREC**), STARBAT will process all selection groups. Otherwise, any subsequent selection groups will be bypassed after the first selection group is satisfied.

In the example above, since **COPYREC** is the function, if an input record had a 1 in column 10 and a 2 in column 20 and an A in column 3 and a B in column 6, then the record would be copied, but only column 30 would be warped. If **COPYSOME** or **COPYALL** were used then both column 30 and column 40 would be warped.

Selecting Members by Content

Members can be selected by their contents. Use a **MBR** form of a function name (for example, **COPYMBR**, **PRINTMBR**, **PRINTHEXMBR**, **PRINTCHRMBR** or **UPDATEMBR**) and members that meet the selection criteria will be processed by the function.

The following example will copy all members that have TEST RECORD in position 23 in one or more records:

```
DD01 COPYMBR IF=(23,EQ,C'TEST RECORD')
```

Selecting Members by Name

Members of a PDS can be selected by their name or by a mask that represents their name. The MEMBER and MEMBERS parameter is used to achieve this.

The MEMBER parameter can only be entered once on a control statement. To specify multiple members, use parentheses with MEMBER; use the MEMBERS parameter; or use multiple control statements with MEMBER and the same **DDxx** name.

The following example will copy a single member.

```
DD01 COPYREC MEMBER=MEMBERA
```

The following example will print two different members.

```
DD01 PRINTHEX MEMBER=MEMBERA
DD01 PRINTHEX MEMBER=MEMBERC
```

The following example will print two different members.

```
DD01 PRINTHEX MEMBER=(MEMBERA,MEMBERB)
```

The following example will copy all members whose names begin with STAR.

```
DD01 COPYREC MEMBERS=STAR
```

The following example will print all members whose names begin with STAR and contain Z in position eight.

```
DD01 PRINTHEX MEMBERS=STAR---Z
```

OPTIONS=JCL Processing

STARBAT supports OPTIONS=JCL. This means that STARBAT can determine JCL control statements and continuations; and each statement can be processed logically. STARBAT recognizes DD, EXEC, JCLLIB, JOB, OUTPUT, PROC, SET JCL statements and their logical continuations.

If a line in the member is not one of these statements or its continuation, STARBAT will use normal change and update processing rules for that single line. Following lines can then be searched or changed in either JCL or normal mode. JCL statements must begin with a // in column 1 and comment statements are considered a part of the concatenation.

Thus, the following would be considered a single statement:

```
//STEPABC  EXEC  PGM=STARBAT,PARM='BATCH',
//* THIS IS A COMMENT IN THE MIDDLE OF A CONTINUED STATEMENT
//  TIME=4,REGION=4096K
```

To search a JCL structure, use the IF or OR statements. You can also search and update JCL statements with the CHANGE, CHANGEALL, OVERLAY and OVERALL statements. Be sure to use a length of zero for the search length of these parameters if you wish to search the entire JCL statement.

The OVERLAY and OVERALL parameters update data but do not shift data columns so they can be used without problems by any of the COPY operations, MULTICOPY or the UPDATE operations.

CHANGE and CHANGEALL can cause data shifting if the search and replacement strings differ in length. If the string is shortened, STARBAT follows normal JCL rules and is able to update any of the COPY, MULTICOPY or UPDATE operations. If the string is expanded, STARBAT adjusts the JCL statement to fit the statement in the new string using normal JCL rules.

JCL expansion is performed using the following logic to minimize changes in JCL statements:

1. If the string will fit, the JCL text is expanded to the right on the changed JCL line.
2. If it does not fit and the string will fit, the JCL text is added after shifting the changed JCL line left and right.
3. If both of the above expansions fail, the JCL line to be changed is split at the previous comma and continued on another line and the added line is expanded as needed. The new JCL line is indented to match the split JCL line.
4. If the above expansion fails because there is not a previous parameter, STARBAT scans for the next comma to the right and splits the JCL statement after that parameter. Again, the new JCL line is indented to match the split JCL line.

STARBAT can not split a JCL line for UPDATE operations. In this case, an expansion error will be indicated and the return code for the job will be set to eight.

Restrictions:

1. A JCL statement may contain up to 50 JCL lines.
2. OPTIONS=JCL and MOVE from input to output is not supported.
3. The JCL is assumed to be syntactically correct; however, STARBAT does not issue syntax messages for incorrect JCL.
4. To be considered JCL, the statements must reside in a PDS member and the data set must have a fixed record format with 80 character records.
5. Only data between columns 1 through 71 are considered for determining control statement boundaries and continuations.

STARBAT Examples

The examples in this section illustrate various uses of StarBat functions and parameters.

Example 1: This example will copy all records and test every third record for a 'PBCE' in column 3. If a match is found, 'PBCE' will be replaced with 4 zeroes.

```
DD01 COPYALL SELECT=3,OVERYLAY=(3,EQ,C'PBCE',C'0000')
```

Example 2: This example will replace 'VOL' with 3 blanks if 'DUMP' is found anywhere in columns 1 through 6 AND if 'VOL' is found 6 relative positions from the location of the character 'D' in 'DUMP'. The record will then be copied to the output data set.

```
DD01 COPYREC IF=(1,6,C'DUMP'),IF=(+6,EQ,C'VOL'),OVERLAY=(+0,3C' ')
```

Example 3: This example will build new records by moving data with MOVE parameters. The first 5 records are printed in hexadecimal format.

```
DD01 COPYREC MOVE=(1,10,5),MOVE=(11,10,20),PRINTHEX=5
```

Example 4: The first DD01 control statement will copy records until 'VOL001' is found. The second DD01 control statement begins with the the 'VOL001' record and stops copying when 'VOL300' is found.

```
DD01 COPYREC STOPIF=(1,EQ,C'VOL001')
DD01 COPYREC STOPIF=(1,EQ,C'VOL300')
```

Example 5: This example will add one year to the date with format YYMMDD in the specified position if a record contains 'APP1' or 'APP3' in the first 9 positions. Only these selected records are written to the output data set.

```
DD01 COPYREC RDW=3,MAXDATERR=100,PIVOTYR=70,
       IF=(1,9,C'APP1'),
           OR=(1,9,C'APP3'),
       WARP=(10,C,YYMMDD,ADD=1Y)
```

Example 6: This example will copy records containing an 8 or 9 and will warp each record with YYDDD or YYMMDD data format. If a record has both formats, only the YYDDD format is warped.

```
DD01 COPYREC IF=(12,EQ,C'8'),
             WARP=(16,C,YYDDD,ADD=5Y),
             IF=(22,EQ,C'9'),
             WARP=(26,C,YYMMDD,ADD=5Y)
```

Example 7: This example will copy all records and warp all 8 and 9 dates.

```
DD01 COPYALL IF=(12,EQ,C'8'),
             WARP=(15,C,YYDDD,ADD=5Y),
             IF=(22,EQ,C'9'),
             WARP=(26,C,YYMMDD,ADD=5Y)
```

Example 8: This example will copy records containing an 8 or 9 and warp all dates on these records.

```
DD01 COPYSOME IF=(12,EQ,C'8'),
              WARP=(16,C,YYDDD,ADD=5Y),
              IF=(22,EQ,C'9'),
              WARP=(26,C,YYMMDD,ADD=5Y)
```

Example 9: Any record containing a '111' anywhere in the record will be replaced with 'ZZZ'. Any record containing a 'DDD' in column 1 will be replaced with a '---'. Any record containing 'ZZZZ' in column 1 will not be written to the output file.

```
DD01 EXCLUDEREC OVERALL=(1,0,C'111', C'ZZZ'),
                OVERLAY=(1,EQ,C'DDD',C'---'),
                IF=(1,EQ,C'ZZZZ'),MAXRECOUT=0
```

Example 10: This example will select every fifth record, and, if it contains a '241104' or a '181022' in position 25, will output the record. A maximum of 5 records can be written to the output data set identified by DD5.

```
DD01 MULTICOPY SELECT=5,IF=(25,EQ,C'241104,181022'),WRITE=DD5,MAXRECOUT=5
```

Example 11: This example will write all records having '011' in column 9 and '01' in column 1 to the data set identified by DD OUT1. It will write all records having '011' in column 9 and not having a "U" or a "V" in column 1 to the data set identified by DD OUT2. It will write all records not having a '011' in column 9 to OUT2.

```
DD01 MULTICOPY IF=(9,EQ,C'011'),IF=(1,EQ,C'01'),WRITE=OUT1,
                   IF=(9,EQ,C'011'),IF=(1,NE,X'E4,E7'),WRITE=OUT2,
                   IF=(9,NE,C'011'),WRITE=OUT2
```

Example 12: This example will insert a JOBLIB DD statement after the jobcard in all members prefixed with PRS. These members will be written to a new data set identified in the JCL by DD NEWJCL.

```
DD01 MULTICOPY MEMBERS=PRS,OPTIONS=JCL,WRITE=NEWJCL,
               IF=(3,0,C' JOB '),MOVE=(1,80C' '),
               MOVE=(1,C'//JOBLIB DD DISP=SHR'),
               MOVE=(+0,C',DSN=JOBLIB.DSN'),WRITE=NEWJCL
```

Example 13: This example will process three input data sets and write the results to a single output file identified by DD DD02O. Control statement DD02 selects a member called C123.

```
DD01 MULTICOPY WRITE=DD02O
DD02 COPYMBR MEMBER=C123
DD03 MULTICOPY WRITE=DD02O
```

Example 14: If two blanks are found in column 1, they will be changed to '**'. Changed records will be written to the output file identified by OUTPUTA in the JCL. Processing will stop when 'END' is found in column 1.

```
DD01 MULTICOPY CHANGE=(1,EQ,C'  ',C'**'),STOPIF=(1,EQ,C'END'),WRITE=OUTPUTA
```

Example 15: This example will write 'CATALOG NAME=' to the output buffer followed by the 15 bytes 11 relative positions after the 'I' in 'IN-CAT' when 'IN-CAT' is found anywhere in the record. It will write the 15 bytes to the current relative position in the output buffer. Then, it will write the record to the output data set specified by CATREPT.

```
DD01 MULTICOPY PADCHAR=C' ',
                   IF=(1,0,C'IN-CAT'),
                    MOVE=(1,C'CATALOG NAME='),
                    MOVE=(+0,15,+11),
                    WRITE=CATREPT,MAXRECOUT=0
```

Example 16: This example will replace the first occurrence of text string 'CYCLE' with blanks and will print the record if text string '** AAAA' is found anywhere in a record.

```
DD01 PRINTCHR IF=(1,0,T'** AAAA'),
                 OVERLAY=(1,0,T'cycle',5C' '),MAXRECOUT=0
```

Example 17: This example will list the contents of all members in the partitioned data set that match the member mask specified. For example, members that match the member mask include TEST111 and TESTAB1.

```
DD01 PRINTCHRMBR MEMBERS=TEST--1
```

Example 18: This example will process the input data set twice, once for each PRINTHEX control statement. Records containing 'EDIT' or 'A99' in position 19 will be printed in hexadecimal format.

```
DD01 PRINTHEX OPTIONS=MULTI,IF=(19,EQ,C'EDIT')
DD01 PRINTHEX IF=(19,EQ,C'A99')
```

Example 19: This example will change the 'SYS001' to a 'SYSTST' for members containing a 'SYS001' anywhere in the first 20 positions of the record.

```
DD01 UPDATEMBR CHANGE=(1,20,'SYS001',C'SYSTST')
```

Example 20: This example will change all occurrences of hex '98365C' to hex '99365C' if a hex '98365C' is found anywhere in the record. The second IF parameter statement is logically OR'd with the first. In the second IF parameter, if a hex '98000C' is found anywhere in the record, all occurrences of hex '98000C' will be changed to hex '99000C'.

```
DD01 UPDATEREC IF=(1,0,X'98365C'),OVERALL=(1,0,X'98365C',X'98365C')
                    IF=(1,0,X'98000C'),OVERALL=(1,0,X'98000C',X'99000C')
```

Example 21: This example will update the data set in place and replace all occurrences of a '!' or a ':' or a '?' with a blank.

```
DD01 UPDATEREC REPLALL=(1,0,C'!,:,?',C' ')
```

Example 22: This example will skip records 1 though 10, output records 11 through 20, skip records 21 through 25, then output records 26 through 45.

```
DD01 SKIP MAXRECIN=10
DD01 COPYREC MAXRECIN=10,MAXRECOUT=10
DD01 SKIP MAXRECIN=5
DD01 COPYREC MAXRECIN=20,MAXRECOUT=20
```

Example 23: This example will skip the last two records beginning at the end of a sequential data set. Any remaining records that have '01SER20' in column 1 will be printed in reverse order.

```
DD01 SKIPREV MAXRECIN=2
DD01 PRINTREV IF=(1,EQ,C'01SER20')
```

STARBAT Sample Execution

The following example illustrates several STARBAT facilities.

```
//SAMPLEA  JOB ...
//PRINT1 EXEC PGM=STARBAT,TIME=1,REGION=2M
//SYSPRINT DD SYSOUT=(,)
//SYSLIST  DD SYSOUT=(,)
//SYSTOTAL DD SYSOUT=(,)
//DD01      DD DSN=SAMPLE.LIB.PDSE(STARBATT),DISP=SHR
//DD01O     DD DSN=SAMPLE.LIB.ASM(STARBATT),DISP=SHR
//SYSIN     DD *
DD01 COPYSOME  IF=(1,EQ,C'1800'),
            WARP=(6,C,YYDDD,ADD=5Y),PRINTHEX=1,
            SUM=(26,4,C,C'The 1800 totals'),
            IF=(12,EQ,C'1900'),
            WARP=(17,C,YY,ADD=2Y),PRINTHEX=1,
            SUM=(42,3,C)
//
```

Figure 2. Sample STARBAT JCL and Control Statements

```
PDS100I STARBAT/Both -- Version 6.1.0  2000.001

Proprietary software product of SERENA Software
Phone (650)696-1800 Support 696-1850 startool@serena.com
LICENSED TO: StarTool will not work after Dec 31, 1998
             To extend, contact SERENA at (650)696-1800
All other rights reserved - use of this software
product by unauthorized persons is strictly prohibited.

DD01 COPYSOME  IF=(1,EQ,C'1800'),
            WARP=(6,C,YYDDD,ADD=5Y),PRINTHEX=1,
            SUM=(26,4,C,C'The 1800 totals'),
            IF=(12,EQ,C'1900'),
            WARP=(17,C,YY,ADD=2Y),PRINTHEX=1,
            SUM=(42,3,C)
 *** End of control statement

PDS220I //DD01      DD  DSN=SAMPLE.LIB.PDSE,DISP=SHR,UNIT=3380,
PDS220I //  DCB=(RECFM=FB,LRECL=80,BLKSIZE=11440,DSORG=PX),VOL=SER=SER003,
PDS220I //  SPACE=(CYL,(420,20))                 /*FREE TRK=1029*/

STRB01I BSAM input is in use
STRB05I DDNAME=SYS00001 DSN=SAMPLE.LIB.ASM opened for BSAM output
STRB05I DCB=(RECFM=FB,LRECL=80,BLKSIZE=11440),VOL=SER=SER003
```

Figure 3. Sample SYSPRINT Output, Part 1

```
--- STRB75E WARP= Invalid character number at column 0006; hex=X'F6F9F1C1F3';
                                                 char=C'691A3'; at record 7
Text:             WARP=(6,C,YYDDD,ADD=5Y)

-- STRB80E SUM= invalid numeric character; char=C'00X0'; at record 7
TEXT:             SUM=(26,4,C,C'The 1800 totals')


Actions taken for:       IF=(1,EQ,C'1800')               --------7 --------4
Actions taken for:  WARP=(6,C,YYDDD,ADD=5Y)              --------4 --------3
Actions taken for:  SUM=(26,4,C,C'The 1800 totals')      --------4 --------3
Actions taken for:  IF=(12,EQ,C'1900')                   --------6 --------3
Actions taken for:  WARP=(17,C,YY,ADD=2Y)                --------3 --------3
Actions taken for:  SUM=(42,3,C)                         --------3 --------3
```

Figure 4. Sample SYSPRINT Output, Part 2

```
STARBAT Version 6.1.0;  Jan  5, 2000;  DD01=SAMPLE.LIB.PDSE(STARBATT)       VOL=SER003  PAGE   1

RECORD      2 DATA    80   CHAR 1800+74123-1900-98-2----+0034+----+----4-123+----5...7----+----8
   ***C H A N G E D***     ZONE FFFF4FFFFF6FFFF6FF6F66664FFFF4666646666F6FFF46666F666F666646666F
                           NUMR 1800E7412301900098020000E0034E0000E000040123E0000500070000E00008
                                ----+----1----+----2----+----3----+----4----+----5...7----+----8

RECORD      2 DATA    80   CHAR 1800+74123-1900-00-2----+0034+----+----4-123+----5...7----+----8
   ***C H A N G E D***     ZONE FFFF4FFFFF6FFFF6FF6F66664FFFF4666646666F6FFF46666F666F666646666F
                           NUMR 1800E7412301900000020000E0034E0000E000040123E0000500070000E00008
                                ----+----1----+----2----+----3----+----4----+----5...7----+----8

RECORD      7 DATA    80   CHAR 1800+691A3-1900-A8-2----+00X0+-*ERROR--4-00X+----5...7----+----8
   *****E R R O R*****     ZONE FFFF4FFFCF6FFFF6CF6F66664FFEF465CDDDD66F6FFE46666F666F666646666F
                           NUMR 1800E6911301900018020000E0070E0C599690040007E0000500070000E00008
                                ----+----1----+----2----+----3----+----4----+----5...7----+----8

RECORD      7 DATA    80   CHAR 1800+691A3-1900-A8-2----+00X0+-*ERROR--4-00X+----5...7----+----8
   *****E R R O R*****     ZONE FFFF4FFFCF6FFFF6CF6F66664FFEF465CDDDD66F6FFE46666F666F666646666F
                           NUMR 1800E6911301900018020000E0070E0C599690040007E0000500070000E00008
                                ----+----1----+----2----+----3----+----4----+----5...7----+----8
```

Figure 5. Sample SYSLIST Output

```
SYSTOTAL Report

                                Following totals developed from
                                SAMPLE.LIB.PDSE  VOL=SER003

'The 1800 totals                                  ----------           155
            SUM=(42,3,C)                          ----------           458
```

Figure 6. Sample SYSTOTAL Output

STARBAT Messages and Codes

STARBAT issues the following return codes to indicate the results of job execution:

Return Code	Description
0	All functions completed successfully
4	An error was found in decoding a control card; all control statement processing is terminated.
8	No user records were copied to an output data set during a **COPYREC**, **COPYALL**, **COPYMBR**, **COPYREV**, **EXCLUDEREC** or **MULTICOPY** function.
12	An input or output master file could not be opened; STARBAT terminates all processing.

STARBAT issues many STRBnnx and PDSnnnx format messages that can be found in the STARWARP **Messages** manual. The following unnumbered messages are unique to STARBAT and are documented here.

Unnumbered Messages

***** End of control statement**
The last parameter on a function has been input; the function can now be executed.

***** Last of control statements**
The last input record has been input in the control statement data set; the end of file was reached.

***** EXCLUDEREC= parameter has been satisfied**
The EXCLUDEREC parameter has been satisfied; this **EXCLUDEREC** function is complete.

***** MAXRECIN= parameter satisfied**
The MAXRECIN parameter has been satisfied; this function is complete.

***** MAXRECOUT= parameter has been satisfied**
The MAXRECOUT parameter has been satisfied; this function is complete.

***** STOPIF= parameter has been satisfied**
The STOPIF parameter has been satisfied; this function is complete

***** MAXDATERR= parameter has been satisfied**
Too many invalid dates have been encountered; this function is complete

***** No //SYSIN DD statement**
No SYSIN DD statement was entered; all corresponding **`//DDxx`** data sets will be copied to **`//DDxxO`** data sets.

***** No //SYSIN DD controls**
The SYSIN DD data set contained no STARBAT control statements; all corresponding **`//DDxx`** data sets will be copied to `//DDxxO` data sets.

`DDxx COPYREC (generated statement)`
Matching **`//DDxx`** and **`//DDxxO`** data sets have been found and the input data set is being copied to the output data set.

Index

A

AB parameter, 18
ABEND parameter, 18
action parameters, 18
aging dates, 33
AND conditions, 43
AND parameter, 18, 21

B

binary data, 8
bit operations, 26

C

C parameter, 19
CA function, 12
CA parameter, 19
CHANGE parameter, 19
CHANGEALL parameter, 19
CM function, 12
CO parameter, 20
concatenated data sets, 5
control parameters, 18
control statements, 7
converting currencies, 31
COPYALL function, 12
COPYMBR function, 12
COPYOVER parameter, 20
COPYREC function, 12
COPYREV function, 12
COPYSOME function, 12
CPR function, 12
CR function, 12
CS function, 12

D

data
  binary, 8
  hexadecimal, 8
  packed, 8
data set identifier, 7
data set types, 5
data sets
  concatenated, 5
  JCL, 6
direct data sets, 5

E

Euro conversion, 31
EX parameter, 20
EXCLUDEREC function, 13
EXCLUDEREC parameter, 20
EXPAND parameter, 20

F

functions, 3, 7, 11
  CA, 12
  CM, 12
  COPYALL, 12
  COPYMBR, 12
  COPYREC, 12
  COPYREV, 12
  COPYSOME, 12
  CPR, 12
  CR, 12
  CS, 12
  EXCLUDEREC, 13
  MC, 13
  MULTICOPY, 13
  P, 14
  PA, 14
  PC, 14
  PCA, 14
  PCM, 14
  PCR, 14
  PH, 15
  PHA, 15
  PHM, 15
  PHR, 15
  PM, 14
  PRINT, 14
  PRINTALL, 14
  PRINTCHR, 14
  PRINTCHRALL, 14
  PRINTCHRMBR, 14
  PRINTCHRREV, 14
  PRINTHEX, 15
  PRINTHEXALL, 15
  PRINTHEXMBR, 15
  PRINTHEXREV, 15
  PRINTMBR, 14
  PRINTREV, 14
  PRR, 14
  S, 15
  SKIP, 15
  SKIPREV, 15
  SKR, 15
  T, 16
  TOTAL, 16
  UA, 16
  UM, 16
  UPDATEALL, 16
  UPDATEMBR, 16

UPDATEREC, 16
UR, 16
XR, 13

H

hexadecimal data, 8

I

IF parameter, 18, 21
ISAM data sets, 5

J

JCL, 6
JCL processing, 24, 45

L

license, ii
limit parameters, 18
logic, 36
logical AND, 43
logical OR, 43

M

M parameter, 22
MAXDATERR parameter, 22
MAXRECIN parameter, 22
MAXRECOUT parameter, 22
MC function, 13
MEMBER parameter, 22
MEMBERS parameter, 23
MOVE parameter, 23
MRI parameter, 22
MRO parameter, 22
MS parameter, 23
MULTICOPY function, 13
MV parameter, 23

N

NEWMBR parameter, 24
NEWMBRS parameter, 24
NM parameter, 24
NMS parameter, 24
numeric, 18

O

OA parameter, 27
OL parameter, 26
OP parameter, 24
OPTIONS parameter, 24
OPTIONS=JCL, 24, 45
OR conditions, 43
OR parameter, 25
OVERALL parameter, 27
OVERLAY parameter, 26

P

P function, 14
P parameter, 27
PA function, 14
packed data, 8
PAD parameter, 27
PADCHAR parameter, 27
parameter grouping, 18
parameters, 3, 4, 7, 17
AB, 18
ABEND, 18
action, 18, 43
AND, 18, 21
C, 19
CA, 19
CHANGE, 19
CHANGEALL, 19
CO, 20
control, 18, 43
COPYOVER, 20
EX, 20
EXCLUDEREC, 20
EXPAND, 20
IF, 18, 21
limit, 18, 43
M, 22
MAXDATERR, 22
MAXRECIN, 22
MAXRECOUT, 22
MEMBER, 22
MEMBERS, 23
MOVE, 23
MRI, 22
MRO, 22
MS, 23
MV, 23
NEWMBR, 24
NEWMBRS, 24
NM, 24
NMS, 24
OA, 27
OL, 26
OP, 24
OPTIONS, 24
OR, 25
OVERALL, 27
P, 27
PAD, 27
PADCHAR, 27
PC, 28
PH, 28
PL, 28
print, 18, 43
PRINT, 27
PRINTCHR, 28
PRINTHEX, 28
PRINTLPI, 28

RBA, 28
RDW, 29
S, 29
SELECT, 29
selection, 18, 43
SK, 29
ST, 30
STARTKEY, 29
STOPIF, 30
SUM, 30
W, 41
WARP, 31
WARPDEF, 38
WD, 38
WRITE, 41
XR, 20
partitioned data sets, 5
PC function, 14
PC parameter, 28
PCA function, 14
PCM function, 14
PCR function, 14
PH function, 15
PH parameter, 28
PHA function, 15
PHM function, 15
PHR function, 15
PL parameter, 28
PM function, 14
PRINT function, 14
PRINT parameter, 27
print parameters, 18
PRINTALL function, 14
PRINTCHR function, 14
PRINTCHR parameter, 28
PRINTCHRALL function, 14
PRINTCHRMBR function, 14
PRINTCHRREV function, 14
PRINTHEX function, 15
PRINTHEX parameter, 28
PRINTHEXALL function, 15
PRINTHEXMBR function, 15
PRINTHEXREV function, 15
PRINTLPI parameter, 28
PRINTMBR function, 14
PRINTREV function, 14
PRR function, 14

R

RBA parameter, 28
RDW parameter, 29
relative location, 7
restrictions, 13, 18
Return code, 53

S

S function, 15
S parameter, 29
sample execution, 51
SELECT parameter, 29
selection parameters, 18
sequential data sets, 5
SERENA, ii
SK parameter, 29
SKIP function, 15
SKIPREV function, 15
SKR function, 15
ST parameter, 30
STARBAT, 3
data set identifier, 7
examples, 47
functions, 3, 7, 11
license, ii
logic, 43
messages, 53
parameters, 3, 4, 7, 17
restrictions, 13, 18
sample, 51
start location, 7
STARTKEY parameter, 29
STOPIF parameter, 30
SUM parameter, 30

T

T function, 16
TOTAL function, 16

U

UA function, 16
UM function, 16
UPDATEALL function, 16
UPDATEMBR function, 16
UPDATEREC function, 16
UR function, 16

V

valid numeric, 18
VSAM data sets, 5

W

W parameter, 41
WARP logic, 36
WARP parameter, 31
WARPDEF parameter, 38
warping dates, 33
warping numeric data, 31
WD parameter, 38
WRITE parameter, 41

X

XR function, 13
XR parameter, 20